# zopetoolkit Documentation

***Release 2.0***

**Zope Foundation and Contributors**

**Jul 28, 2017**

# Contents

The Zope Toolkit (ZTK) is a set of libraries intended for reuse by projects to develop web applications or web frameworks. It is developed by the contributors of the Zope Foundation.

The whole collection of ZTK libraries are used in various web frameworks and web application servers, two examples of these are Grok and Zope. If you install one of these systems, you will get a part of the ZTK along with it automatically.

CHAPTER 1

# Installation

The Zope Toolkit cannot be installed directly except as individual libraries (such as `zope.component`). To install it you typically would install a framework or application that makes use of these libraries.

If you want to use the Zope Toolkit Known-Good-Set (KGS), you should copy the ztk-versions.cfg file into your own project and include it in your buildout via the extends mechanism.

Frameworks and applications have their own set of install instructions. You should follow these in most cases.

CHAPTER 2

# Python versions

The ZTK supports CPython 2.7, 3.4, 3.5, 3.6 and PyPy.

CHAPTER 3

# Documentation

## About the Zope Toolkit

The Zope Toolkit consists of a set of libraries, which are commonly used by multiple web frameworks (e.g. Grok).

Only a small subset of libraries from the Zope community are part of the Toolkit. The Toolkit is in principle not limited to libraries from Zope community.

### Toolkit libraries

The Toolkit libraries are standalone libraries which have their own releases, bug trackers and in some cases active developer communities around them. The toolkit acts as an umbrella project to deal with aspects which span multiple of these libraries or have impacts on the dependent frameworks.

The set of libraries that is part of the Toolkit can change over time depending on how these libraries evolve and are used. New libraries considered for inclusion can be added to the set, and existing libraries no longer used can be removed from the set.

So the set of Zope Toolkit libraries is not static; what is included continuously evolves. The project maintains a list of which libraries are part of the Toolkit.

### Community libraries

Surrounding the Zope Toolkit libraries a large number of other libraries exists. These libraries might integrate with the Zope Toolkit and might make use of the Zope Toolkit.

## Development process

This area covers the procedures and guidelines we adhere to when developing Zope Toolkit components.

## Bug tracking

ZTK bugs are tracked in individual projects (one per package) using Github's bug trackers.

`XXX`, `FIXME` or `TODO` comments in the source code are not considered bugs and therefore cannot block releases. Consider adding a bug in addition to the comment. If something is really broken and you want to indicate it with `XXX`, consider instead not checking this into the trunk of a project but into a separate working branch.

### Historical bugs

Historically, bugs have been tracked with the Zope 3 project (https://bugs.launchpad.net/zope3).

Even longer ago, Zope 3 used its own bug tracker (which lived at http://www.zope.org/Collectors). Those were imported into the Launchpad bug database and have aliases of the form "zope3dev-NNN" where "NNN" is the old bug number, so they can be retrieved by searching for these on Launchpad.

## Automated test suite / nightly builds

The ZTK builds on the automated test suites (unit and functional tests) from the individual projects it keeps track of. We use automated build systems, like travis-ci, to run various combinations of differing Python versions, operating systems and packages and ensure everything works as expected.

### The automated test suite

The ZTK's automated test suite builds on the individual packages' unit and functional tests and creates a combined test runner that runs each packages' test suite in isolation but ensures that the dependencies are satisfied using the ZTK versions under test.

The combined test runner is created using z3c.recipe.compattest – check the documentation for details.

If you take a ZTK checkout, you can run the tests yourself like this:

```
$ git clone git@github.com:zopefoundation/zopetoolkit.git
$ python bootstrap.py
$ bin/buildout
$ bin/test-ztk
```

If you work on a ZTK package and want to ensure that your changes are compatible with all other ZTK libraries, you can use a checkout of the individual package inside the zopetoolkit checkout:

```
$ bin/develop co zope.component
$ bin/develop rb
$ bin/test-ztk
```

The develop commands get a git checkout of the specified package and puts it into the develop/ folder, so you make your changes there.

## Releasing software

When releasing software, the following steps should be taken:

1. Make sure all automated tests of the package pass and there are no release-critical bugs in its bug tracker.

2. Fill in the release date in `CHANGES.rst`. Make sure the changelog is complete. Update the version number in `setup.py` and remove the `dev` marker.

   Also consider the version number for the new release. If the API has been added to, or the behavior has otherwise changed in a way that goes beyond a bugfix, please update the version number for the next release and make it a feature release (x.y) instead of a bugfix release (x.y.z).

3. Make sure the package metadata in `setup.py` is up-to-date. You can verify the information by re-generating the egg info:

   ```
   python setup.py egg_info
   ```

   and inspecting `src/EGGNAME.egg-info/PKG-INFO`. You should also make sure the that the long description renders as valid reStructuredText. You can do this by using the `rst2html.py` utility from docutils:

   ```
   python setup.py --long-description | rst2html.py > test.html
   ```

   If this will produce warnings or errors, PyPI will be unable to render the long description nicely. It will treat it as plain text instead.

4. Create a release tag.

5. Get a separate checkout of the release tag for creating the distribution tarball and eggs. It is important that you don't do this on the master or release branch to avoid

   - forgetting to tag the release at all.
   - forgetting to clean up the `build` directory that distutils and setuptools create. Failure to do so may result in old artefacts in eggs.
   - forgetting to check in files that are needed by `setup.py` or as package data. Setuptools will only include them in the distribution if they are specifed by the `MANIFEST.in`.

   In the checkout of the tag create a distribution and upload it to PyPI using the following command:

   ```
   python setup.py register sdist upload
   ```

6. Back on the master or the release branch, increase the version number in `setup.py` to the *next* release and add back the `dev` marker. The convention is that the master or release branch always points to the upcoming release, *not* the one that has been released already. So if you've just released version 3.4.1, you should change `setup.py` to read:

   ```
   setup(
       name='...',
       version='3.4.2.dev',
       ...
       )
   ```

   In `CHANGES.rst` add a *new* section for the upcoming release. The release date for that should say "unreleased" so that committers recording their changes won't accidentally put their entry in the section for an already released version. For example:

   ```
   3.4.2 (unreleased)
   ------------------

   * ...

   3.4.1 (2007-01-24)
   ------------------
   ```

```
* Fixed bug in the foo adapter.

* Added a bar utility for optimized kaboodling.

3.4.0 (2006-09-13)
------------------

Initial release as separate egg.
```

**Important:** Once released to PyPI or any other public download location, a release may *never* be removed, even if it has proven to be a faulty release ("brown bag release"). In such a case it should simply be superseded immediately by a new, improved release.

## Maintaining software

**Author** Philipp von Weitershausen <philipp@weitershausen.de>

**Status** Draft

**Contents**



### Introduction

This guide outlines rules, recommendations and best practices for authors of software (mainly Python packages) that lives in the Zope software repository. It does not impose any new rules on existing software. It is mostly a written-down collection of rules, recommendations and best practices that have been maintained within the Zope community for some time now, some more than others.

Therefore, the main purpose of this document is to document those existing practices for people who've joined the Zope project recently, and to serve as a canonical source for guidance when in doubt.

### Repository layout

Here's an example of naming conventions used in the Zope Github organization at `github.com/zopefoundation`:

```
zope.component/
zope.component/branches/
zope.component/branches/3.4
```

```
zope.component/branches/philikon-100x-faster
...
zope.component/tags/
zope.component/tags/3.4.0
zope.component/tags/3.4.0b1
zope.component/tags/3.4.1
...
zope.component/master/
...
```

To summarize:

- The Github project name is the project's name. This is also the name of the Python distribution. If it contains just one package, the dotted name of the package should be used for the project name, e.g. `zope.component`, `z3c.form`. The same is true for Zope 2 "products", e.g. `Products.Five` (note that not all products adhere to this for legacy reasons, new projects should use this convention, however).

  It is recommended to put software in a namespace package to avoid name clashes. Valid choices for namespace package names are:

  - `zope`, although this one is used by the Zope platform itself and should only be used for new projects after approval from the community.
  - `z3c` (meaning "Zope 3 Community")

  Sometimes the project doesn't hold just one package but a number of packages or not even software at all. In this case pick a meaningful name that's unlikely to interfere with other names.

- Release branches and release tags should be simple dotted numbers stating the version that the branch or tag is for. Use `3.4`, not `3.4.x` or something else, for the release branch that's responsible for `3.4.0`, `3.4.1`, etc.

- Branches for doing temporary work (such as refactorings) should begin with the login name of the primary author and then feature a short and descriptive name of what they're about. For example, `philikon-100x-faster` indicates that Philipp von Weitershausen is working in making `zope.component` 100 times faster. You wish...

  Temporary work branches should be short-lived and *removed* once they're merged. Release branches and tags should never be removed. Release tags shouldn't be committed to once the release has been announced and distributed.

### License

Unless allowed otherwise, all software committed to the Zope organization is subject to the Zope Public License (ZPL). The documentation of the software should state so. In addition, each source code file should contain the following license header at the top:

```
##############################################################################
#
# Copyright (c) 2016 Zope Foundation and Contributors.
# All Rights Reserved.
#
# This software is subject to the provisions of the Zope Public License,
# Version 2.1 (ZPL).  A copy of the ZPL should accompany this distribution.
# THIS SOFTWARE IS PROVIDED "AS IS" AND ANY AND ALL EXPRESS OR IMPLIED
# WARRANTIES ARE DISCLAIMED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
# WARRANTIES OF TITLE, MERCHANTABILITY, AGAINST INFRINGEMENT, AND FITNESS
# FOR A PARTICULAR PURPOSE.
#
##############################################################################
```

The year and the current version of the ZPL are to be adjusted, of course.

### Coding style

When starting new packages, one should adhere to the coding style suggested by PEP 8. When modifying or enhancing existing software, the package's existing coding style should be used.

### Automated tests

All software should be accompanied by automated tests. Packages that provide integrated components for the web should preferrably be accompanied by both unit tests and integration/functional tests. The definition of test cases should be done in `tests` packages or modules.

Before checking modifications into the trunk or a release branch, all existing tests for the package must pass. Furthermore, when adding a feature, modifying the behaviour of a component or fixing a bug, a test exercising the change must be supplied as well. There would otherwise be no reproducible way of knowing whether the new code actually worked. In terms of automated tests, think "Untested code is broken code."

Tests should be written in a fairly literate way with documentation of the test itself. That is to ensure that the intent of each test is clear and obvious to any other developer. One should use `unittest.TestCase` as a test harness.

### Backward-compatibility

As a rule of thumb, backwards-incompatible changes to stable, released code should be avoided. Examples of backwards-incompatible changes are

- renaming packages, modules, classes, functions, etc. without ensuring the old import paths still work,
- changing a public interface, which also includes *adding* attributes or methods (imagine people implemented this interface in their own code, now all of a sudden their implementations don't comply with the interface anymore)

If you'd like to replace a certain component or package with another, better one, don't remove the original component or package, not even after a deprecation period. Instead, consider simply abandoning the original component or package. You should clearly document that, of course, possibly even by raising DeprecationWarnings. Then you provide the replacement under a separate name.

For example, consider you would like to radically improve a package `mycorp.foo`. Instead of changing it in an incompatible way, you should just stop supporting it and create `mycorp.newfoo` (or whatever you'd like to name it).

Consistency weighs higher than cleanliness.

### Package documentation and metadata

It is recommended that all packages in the Zope repository are accompanied by at least the following minimum set of documentation and metadata (file names are relative to the package's distribution, in terms of a checkout they're relative to `master` or a release branch or tag):

**`README.rst`** This file should give an overview over what the package or project is about. It is acceptable for this to be just a few paragraphs or a full-fledged manual for the piece of software.

If the package has an associated mailinglist and a bugtracker, it is a good idea to mention it here.

This file should contain valid reStructuredText.

Here's an example for a short file containing only a few paragraphs, but referring to a separate documentation site:

```
Martian provides infrastructure for declarative configuration of
Python code. Martian is especially useful for the construction of
frameworks that need to provide a flexible plugin infrastructure.

Martian provides a framework that allows configuration to be expressed
in declarative Python code. These declarations can often be deduced
from the structure of the code itself. The idea is to make these
declarations so minimal and easy to read that even extensive
configuration does not overly burden the programmers working with the
code. Configuration actions are executed during a separate phase
("grok time"), not at import time, which makes it easier to reason
about and easier to test.

For more information about using Martian, see:

  martian.readthedocs.io
```

**CHANGES.rst** This file contains the changelog. The changelog should keep track of every new feature and every bugfix of all releases. When a particular release has lots of changes, it may group them into "Features" and "Bugfixes". The release date should be given for each release in the ISO 8601 dash notation (YYYY-MM-DD). For example:

```
1.1 (unreleased)
----------------

* ...

1.0 (2007-01-24)
----------------

* Fixed a memory leak.

* Improved documentation a lot.

0.9 (2006-12-05)
----------------

* Initial preview release.
```

This file should contain valid reStructuredText.

**setup.py** Most Python software is distributed using distutils and setuptools. By convention, the script to do the packaging should be called `setup.py`. The following example outlines the *minimum* package metadata that it should contain:

```
from setuptools import setup, find_packages

long_description = (open('README.txt').read() +
                    '\n\n' +
                    open('CHANGES.txt').read())

setup(
    name='z3c.awesomelib',
    version='2.0.0.dev',
    url='http://pypi.python.org/pypi/z3c.awesomelib',
```

```
    author='Philipp von Weitershausen',
    author_email='philipp@weitershausen.de',
    license='ZPL 2.1',
    classifiers=['Environment :: Web Environment',
                 'Intended Audience :: Developers',
                 'License :: OSI Approved :: Zope Public License',
                 'Programming Language :: Python',
                 'Operating System :: OS Independent',
                 'Topic :: Internet :: WWW/HTTP',
                 ],
    description="An awesome website implementation.",
    long_description=long_description,

    packages=find_packages('src'),
    package_dir={'': 'src'},
    namespace_packages=['z3c'],
    include_package_data=True,
    install_requires=['setuptools', 'zope.interface, 'zope.component']
    zip_safe=False,
    )
```

To elaborate on this example:

- The blank line separates mostly informational metadata intended for users from packaging metadata intended for setuptools.
- Many packages don't have their own "homepage". It is often more convenient to use the Python Package Index (PyPI) as a homepage for the package (via the `url` parameter) since PyPI renders `long_description` for the package's main page and provides downloads.
- The list of Trove classifiers (`classifiers` parameter) should be adjusted according to the specific package, of course. Much of the software in the Zope repository is intended to be used with Zope 2 or the Zope Toolkit (sometimes for both), we aim to make more and more software available for independent use (well-known examples are `zope.interface` or the `ZODB`).
- `description` should be a one-sentence description of the package while `long_description` is best taken from the `README.rst` file as demonstrated. You may also include the changelog in `long_description` by concatenating `README.rst and ``CHANGES.rst`.

## Communication

Communication for the Zope Toolkit development itself happens both on IRC and in a mailing list.

### IRC

We use #zope on the IRC network provided by freenode.net.

### Mailing list

The official Zope Toolkit development mailing list is zope-dev@zope.org. You can subscribe to it by visiting http://mail.zope.org/mailman/listinfo/zope-dev.

# Coding style

The coding style is intended to support a consistent code base by laying out rules for how to work on Zope code, for example how to structure files, format your source code and naming things.

## Python

The general rule when writing Python code is to follow PEP 8. The rules given later in this document override what is said in PEP 8.

Be tolerant of code that doesn't follow these conventions: our code base has been evolving over years and doesn't always match the current style as we update these rules. Also, we want to reuse software written for other projects that do not adhere to our rules.

Remember that PEP 8 explicitly allows breaking a rule in the interest of keeping code consistent.

A reasonable goal is that code covered by the ZPL should follow these conventions.

### License statement, module docstring

Python files should always contain the most current license comment at the top followed by the module documentation string.

Here is the template:

```
##############################################################################
#
# Copyright (c) 2016 Zope Foundation and Contributors.
# All Rights Reserved.
#
# This software is subject to the provisions of the Zope Public License,
# Version 2.1 (ZPL).  A copy of the ZPL should accompany this distribution.
# THIS SOFTWARE IS PROVIDED "AS IS" AND ANY AND ALL EXPRESS OR IMPLIED
# WARRANTIES ARE DISCLAIMED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
# WARRANTIES OF TITLE, MERCHANTABILITY, AGAINST INFRINGEMENT, AND FITNESS
# FOR A PARTICULAR PURPOSE
#
##############################################################################
"""One-line summary goes here.

Module documentation goes here.
"""
```

### Whitespace

Trailing whitespace should not occur, nor should multiple blank lines at the end of files.

### Import statements

All imports should be at the top of the module, after the module docstring and/or comments, but before module globals.

It is sometimes necessary to violate this to address circular import problems. If this is the case, add a comment to the import section at the top of the file to flag that this was done.

Refrain from using relative imports. Instead of:

```
import .foo  # from same package
```

you can write:

```
from zope.package.module import foo
```

### Attribute and method names

The naming of attributes as recommended by PEP 8 is controversial. PEP 8 prefers `attribute_name` over `attributeName`. Newer code tends to prefer the use of underscores over camel case. However, Zope has been built originally with the latter rule and a lot of code still use this scheme.

Boolean-type attributes should always form a true-false-question, typically using "has" or "is" as prefix. Example: `is_required` instead of `required`.

Method names should always start with a verb that describes the action.

Examples:

```
good:
first_name
is_required
execute_command()
save()
convert_value_to_string()

bad:
FirstName
required
command()
string()
```

### Local variables

Single-letter variable names should be avoided unless:

- Their meaning is extremely obvious from the context, and
- Brevity is desirable

The most obvious case for single-letter variables is for iteration variables.

### `try`/`except` blocks

`try` blocks should cover as little code as possible. `except` statements should match exceptions as specific as possible.

For example, if you are converting a value to an `int`, and you want to catch conversion errors, you need only catch `ValueError`. Be sure to do the minimum possible between your `try:` and `except ValueError:` statements:

```
try:
    int(x)
except ValueError:
    ...
```

### String handling

Use `startswith` and `endswith` because it is faster, cleaner and less error-prone than comparing sliced strings:

```
# Yes:
if foo.startswith('bar'):
    ...
if foo.endswith('.html'):
    ...

# No:
if foo[:3]=='bar':
    ...
if foo[-5:]=='.html':
    ...
```

### Type checks

Constructs like `if type(obj) is type('')` should be replaced using `isinstance()`:

```
# Yes:
if isinstance(obj, int):
  ...

# No:
if type(obj) is type(1):
  ...
if type(obj) is int:
```

### Marker objects

Use instances of `object` if you need to construct marker objects (for example when detecting default values). Compare them using `is` as recommended by PEP 8.

### Interfaces

Interface names adhere to PEP 8's naming of classes, except that they are prefixed with a capital `I`, as in `IMagicThing`.

One function of interfaces is to document functionality, so be very verbose with the documentation strings.

All public interfaces should go into a file called `interfaces.py`. "Public" interfaces are those that you expect to be implemented more than once. Interfaces that are likely to be implemented only once, like `IGlobalAdapterService`, should live in the same module as their implementation.

## ZCML

### File naming conventions

ZCML configuration for a package should, in general, be placed in a file named `configure.zcml`. If a package performs meta configuration (defines new configuration directives), the meta configuration should go in a file named `meta.zcml`. The code that implements the meta configuration directives should (again, in general) go in a file named `metaconfiguration.py`.

### Style guide

The ZCML style guide has been developed with the following qualities in mind:

- Lines under 80 characters wherever possible
- Indentation to reflect nesting
- Ease of maintenance
- Easy visual scanning through ZCML

### Tabs and spaces

All whitespace to be made up of space characters. No chr(9) hard tabs.

Indentation of 2 characters to show nesting, 4 characters to list attributes on separate lines. This distinction makes it easier to see the difference between attributes and nested elements.

### Logical sections

If a configuration file has many logical sections, then mark the sections with comments and indent the sections relative to the comments. For example:

```
<configure xmlns="http://namespaces.zope.org/zope">

  <!-- Configuration registries -->

  <content
      class="zope.app.services.configuration.ConfigurationRegistry"
      >
    <require
        permission="zope.ManageServices"
        interface="zope.app.interfaces.services.configuration.IConfigurationRegistry"
        />
  </content>

  <!-- Adapter Service -->

  <content class="zope.app.services.adapter.AdapterService">
    <implements

  ...
```

### Namespaces

Always use the "usual" names for namespaces in the document:

**default (no qualification needed)** for http://namespaces.zope.org/zope

**browser** for http://namespaces.zope.org/browser

**zmi** for http://namespaces.zope.org/zmi

**security** for http://namespaces.zope.org/security

**xmlrpc** for http://namespaces.zope.org/xmlrpc

**whatevername** for http://namespaces.zope.org/whatevername

Only define those namespaces used in the document. This is a similar rule to what imports to use in a Python module.

### Opening tags

Close a one-line tag on the same line:

```
<class class="Foo">
```

Close a multi-line tag on a new line, at the same level of indentation as the tags attributes:

```
<browser:pages
    name="index.html"
    class=".index.Index"
    >
```

### Empty tags

Close a one-line tag on the same line, insert a space after the last attribute::

```
<subscriber handler=".foo.OnFoo" />
```

Close a multi-line tag on a new line at the same level of indentation as the tag's attributes:

```
<browser:page
    name="index.html"
    class=".index.Index"
    />
```

### Closing tags

Indent closing tags at the same level of indentation as the opening tags:

```
<class class=".foo.Foo">
    ...
</class>
```

### Attributes

If all the attributes fit on one line with the tag name, do that:

```
<class class=".foo.Foo">
```

If all the attribute fit on one line without the tag name, do that on the line after the tag, indented 4 spaces along from the tag:

```
<browser:page
    name="index.html" class=".foo.Foo" permission="zope.View"
    />
```

Otherwise, put the first attribute on a new line, and use one line per attribute:

```
<browser:page
    name="index.html"
    class=".foo.Foo"
    permission="zope.View"
    template="foo.pt"
    />
```

Use double quotes for attributes unless single quotes are needed to enclose double quotes.

### Comments

Comments should be placed immediately above the declarations they apply to. Keep comments to one line where possible, and open and close the comment on the same line.

## TODO comments

Occasionally you may need to note a place in a file that needs to be revisited later. There are three standard codes used to designate such places: `XXX`, `TODO`, and `BBB`.

Depending on the type of the file, those codes should be placed within a comment according to the applicable syntax.

**XXX** XXX comments should only be used during development to note things that need to be taken care of before a final (trunk) commit:

```
# XXX This will break if someone types `9`.
```

One should not expect to see XXX in released software.

It should be extremely rare to check in an XXX on a trunk. If an XXX is checked in on a trunk:

- **The intention and expectation will be that someone will resolve** the XXX before someone releases the code.
- **The XXX must fully describe an issue, so that someone else can** read the comment and know what it's about.

XXX shall not be used to record new features, non-critical optimization, design changes, etc.

**TODO** If you want to record things like new features, non-critical optimizations, design changes, or other long term changes, use TODO comments:

```
# TODO This could go faster if we'd use a better data structure.
```

People making a release shouldn't care about TODOs, but they ought to be annoyed to find XXXs.

**BBB** When adding code to preserve backward compatibility, use a BBB comment with a date. For example:

```
# BBB 2016-07-08, preserves use of 'get_foo' function
```

## Check-in guidelines

Please be careful before a check in to make sure:

- All the code you are about to check in has reasonable test coverage. If you want to check in partial code in order to collaborate remotely, do so in a branch.

- All the tests pass. Running individual tests for the code you are working on is fine while developing, but immediately before a check in, all tests must be run.

  Build environments based on buildout usually do so by running:

```
bin/test --all
```

- Use 'git status' to make sure you have added any new files to the repository. Alternatively you could make a fresh checkout before running the tests.
- Please be aware of the version of Python you use. The recommended and supported versions of Python vary over time (although they do so slowly).

## Writing tests

For any module 'somepkg.somemod' there should be a corresponding unit test module 'somepkg.somemod.tests.test_somemod'. Or if more than one set of unit tests is desired, multiple test modules of the form 'somepkg.somemod.tests.test_somemodYYYY'. Note that this means that your 'somemod' directory needs to have a 'tests' subdirectory, and that that subdirectory must have a (normally empty) '__init__.py' file in it.

In your unit test class, begin all unit test methods with the string 'test'. If you use the CleanUp class support, make sure that your 'setUp' and 'tearDown' methods call the CleanUp class's 'setUp' and 'tearDown' methods:

```
class TestSomething(CleanUp):
    def setUp(self):
        CleanUp.setUp(self)
        #your setup here

    def tearDown(self):
        #your teardown here
        CleanUp.tearDown(self)
```

**Never** give your test methods a docstring! Doing so makes it very difficult to find your test method when using the verbose output. Use a comment instead.

Call your test class TestSomething, never just Test.

Call your test methods test_something(), not testSomething()

The typical approach to structuring the tests themselves is to write test methods that exercise each individual method of the class. It is a good idea to organize these tests according to the Interfaces implemented by the class under test. In fact, it is often best to implement such tests in separate mixin classes, one class per Interface.

Within the unit tests themselves, the Zope style is to use the positive rather than the double negative assertions. Thus, use `assertEqual` rather than `failUnlessEqual`, `assertRaises` rather than `failUnlessRaises`, and `assert_` rather than `failUnless`. (Yes, `assert_` is an ugly name, but it is still preferred.)

There are certain other "Best Practices" the following of which leads to more robust and general unit tests:

- If at all possible, avoid writing to the file system. It should be possible to run the tests with only read-only access to the test directories. If you do have to create files, create them using the python tempfile module. Be sure to clean up after yourselves in your tearDown method.
- It has been suggested that doing import of the module under test in the global section of the test module is bad, because in some uses of unittest test modules that generate import errors outside the tests themselves are ignored silently. It has also been suggested that if so this is a bug in unittest. I'm not aware of a definitive answer to this question. If you wish to avoid globally importing the module under test, you can write a "helper function" to do the import and create and return instance(s) of the objects needed for testing, and call it from the start of each unit test.

# Attic

This section contains outdated information, archived for historical purposes.

## Releases

This area collects release-specific information about the toolkit including a list of backward-incompatible changes, new techniques developed, and libraries included.

### Zope Toolkit 1.1

This document covers major changes in this release that can lead to backward-incompatibilities and explains what to look out for when updating.

#### Introduction

The Zope Toolkit 1.1 release is the second feature release of the Zope Toolkit. The Zope Toolkit really is just a collection of libraries managed together by the Zope developers. We typically treat each library independently, so you would like to look at the CHANGES.txt in each library for updates. Here we note larger changes, especially ones that affect multiple libraries.

#### Installation

The Zope Toolkit cannot be installed directly except as individual libraries (such as `zope.component`). To install it you typically would install a framework or application that makes use of these libraries. Examples of such projects are BlueBream, Grok and Zope 2.

If you want to use the Zope Toolkit KGS, you can use the buildout extends mechanism (replace 1.1 by the desired version):

```
[buildout]
extends = http://download.zope.org/zopetoolkit/index/1.1/ztk-versions.cfg
```

You can also copy the file locally or additionally extend the zopeapp-versions.cfg file from the same location.

Frameworks and applications have their own set of install instructions. You should follow these in most cases.

#### Python versions

The ZTK 1.1 release series aim to support Python 2.5 up to Python 2.7. However, there is an issue with the *zope.proxy* package using Python 2.7 on 64-bit machines. This issue should be resolved with the upcoming release of Python 2.7.2. See also http://bugs.python.org/issue10360.

Some libraries included in the ZTK support Python 3.1 or later. But as a whole the ZTK supports no Python 3.x version yet.

### News

### Python versions

Python 2.4 is no longer supported by this version of the ZTK. Support for Python 2.7 has been added.

### ZODB 3.10

This ZTK version includes ZODB 3.10 instead of 3.9 as included in the 1.0 series. You can read more about the changes at http://pypi.python.org/pypi/ZODB3/3.10.0#change-history.

### zc.buildout 1.5

The buildout version has been updated to 1.5 from the former 1.4 series. This release requires some changes to recipes, so make sure to update all recipes to compatible versions or check their availability first. More detailed changes can be found at http://pypi.python.org/pypi/zc.buildout/1.5.2#change-history.

### zope.testing 3.10

In zope.testing 3.10 the *zope.testing.testrunner* package has been moved to a standalone distribution called *zope.testrunner*. You need to adjust your imports or use compatible versions of test runner recipes.

### Zope Toolkit 1.1c1

This document covers major changes in this release that can lead to backward-incompatibilities and explains what to look out for when updating.

### Introduction

The Zope Toolkit 1.1 release is the second feature release of the Zope Toolkit. The Zope Toolkit really is just a collection of libraries managed together by the Zope developers. We typically treat each library independently, so you would like to look at the CHANGES.txt in each library for updates. Here we note larger changes, especially ones that affect multiple libraries.

### Installation

The Zope Toolkit cannot be installed directly except as individual libraries (such as `zope.component`). To install it you typically would install a framework or application that makes use of these libraries. Examples of such projects are BlueBream, Grok and Zope 2.

If you want to use the Zope Toolkit KGS, you can use the buildout extends mechanism (replace 1.1 by the desired version):

```
[buildout]
extends = http://download.zope.org/zopetoolkit/index/1.1/ztk-versions.cfg
```

You can also copy the file locally or additionally extend the zopeapp-versions.cfg file from the same location.

Frameworks and applications have their own set of install instructions. You should follow these in most cases.

### Python versions

The ZTK 1.1 release series supports Python 2.5 up to Python 2.7. Some libraries included in the ZTK support Python 3.1 or later. But as a whole the ZTK supports no Python 3.x version yet.

### News

### Python versions

Python 2.4 is no longer supported by this version of the ZTK. Support for Python 2.7 has been added.

### ZODB 3.10

This ZTK version includes ZODB 3.10 instead of 3.9 as included in the 1.0 series. You can read more about the changes at http://pypi.python.org/pypi/ZODB3/3.10.0#change-history.

### zc.buildout 1.5

The buildout version has been updated to 1.5 from the former 1.4 series. This release requires some changes to recipes, so make sure to update all recipes to compatible versions or check their availability first. More detailed changes can be found at http://pypi.python.org/pypi/zc.buildout/1.5.2#change-history.

### zope.testing 3.10

In zope.testing 3.10 the *zope.testing.testrunner* package has been moved to a standalone distribution called *zope.testrunner*. You need to adjust your imports or use compatible versions of test runner recipes.

### Zope Toolkit 1.1.6

This document covers major changes in this release that can lead to backward-incompatibilities and explains what to look out for when updating.

### Introduction

The Zope Toolkit 1.1 release is the second feature release of the Zope Toolkit. The Zope Toolkit really is just a collection of libraries managed together by the Zope developers. We typically treat each library independently, so you would like to look at the CHANGES.txt in each library for updates. Here we note larger changes, especially ones that affect multiple libraries.

### Installation

The Zope Toolkit cannot be installed directly except as individual libraries (such as `zope.component`). To install it you typically would install a framework or application that makes use of these libraries. Examples of such projects are BlueBream, Grok and Zope 2.

If you want to use the Zope Toolkit KGS, you can use the buildout extends mechanism (replace 1.1 by the desired version):

```
[buildout]
extends = http://download.zope.org/zopetoolkit/index/1.1/ztk-versions.cfg
```

You can also copy the file locally or additionally extend the zopeapp-versions.cfg file from the same location.

Frameworks and applications have their own set of install instructions. You should follow these in most cases.

### Python versions

The ZTK 1.1 release series aim to support Python 2.5 up to Python 2.7. However, there is an issue with the *zope.proxy* package using Python 2.7 on 64-bit machines. This issue should be resolved with the upcoming release of Python 2.7.2. See also http://bugs.python.org/issue10360.

Some libraries included in the ZTK support Python 3.1 or later. But as a whole the ZTK supports no Python 3.x version yet.

### News

### Python versions

Python 2.4 is no longer supported by this version of the ZTK. Support for Python 2.7 has been added.

### ZODB 3.10

This ZTK version includes ZODB 3.10 instead of 3.9 as included in the 1.0 series. You can read more about the changes at http://pypi.python.org/pypi/ZODB3/3.10.0#change-history.

### zc.buildout 1.5

The buildout version has been updated to 1.5 from the former 1.4 series. This release requires some changes to recipes, so make sure to update all recipes to compatible versions or check their availability first. More detailed changes can be found at http://pypi.python.org/pypi/zc.buildout/1.5.2#change-history.

### zope.testing 3.10

In zope.testing 3.10 the *zope.testing.testrunner* package has been moved to a standalone distribution called *zope.testrunner*. You need to adjust your imports or use compatible versions of test runner recipes.

### Zope Toolkit 1.1.5

This document covers major changes in this release that can lead to backward-incompatibilities and explains what to look out for when updating.

### Introduction

The Zope Toolkit 1.1 release is the second feature release of the Zope Toolkit. The Zope Toolkit really is just a collection of libraries managed together by the Zope developers. We typically treat each library independently, so you would like to look at the CHANGES.txt in each library for updates. Here we note larger changes, especially ones that affect multiple libraries.

### Installation

The Zope Toolkit cannot be installed directly except as individual libraries (such as `zope.component`). To install it you typically would install a framework or application that makes use of these libraries. Examples of such projects are BlueBream, Grok and Zope 2.

If you want to use the Zope Toolkit KGS, you can use the buildout extends mechanism (replace 1.1 by the desired version):

```
[buildout]
extends = http://download.zope.org/zopetoolkit/index/1.1/ztk-versions.cfg
```

You can also copy the file locally or additionally extend the zopeapp-versions.cfg file from the same location.

Frameworks and applications have their own set of install instructions. You should follow these in most cases.

### Python versions

The ZTK 1.1 release series aim to support Python 2.5 up to Python 2.7. However, there is an issue with the *zope.proxy* package using Python 2.7 on 64-bit machines. This issue should be resolved with the upcoming release of Python 2.7.2. See also http://bugs.python.org/issue10360.

Some libraries included in the ZTK support Python 3.1 or later. But as a whole the ZTK supports no Python 3.x version yet.

### News

### Python versions

Python 2.4 is no longer supported by this version of the ZTK. Support for Python 2.7 has been added.

### ZODB 3.10

This ZTK version includes ZODB 3.10 instead of 3.9 as included in the 1.0 series. You can read more about the changes at http://pypi.python.org/pypi/ZODB3/3.10.0#change-history.

### zc.buildout 1.5

The buildout version has been updated to 1.5 from the former 1.4 series. This release requires some changes to recipes, so make sure to update all recipes to compatible versions or check their availability first. More detailed changes can be found at http://pypi.python.org/pypi/zc.buildout/1.5.2#change-history.

### zope.testing 3.10

In zope.testing 3.10 the *zope.testing.testrunner* package has been moved to a standalone distribution called *zope.testrunner*. You need to adjust your imports or use compatible versions of test runner recipes.

### Zope Toolkit 1.1.4

This document covers major changes in this release that can lead to backward-incompatibilities and explains what to look out for when updating.

### Introduction

The Zope Toolkit 1.1 release is the second feature release of the Zope Toolkit. The Zope Toolkit really is just a collection of libraries managed together by the Zope developers. We typically treat each library independently, so you would like to look at the CHANGES.txt in each library for updates. Here we note larger changes, especially ones that affect multiple libraries.

### Installation

The Zope Toolkit cannot be installed directly except as individual libraries (such as `zope.component`). To install it you typically would install a framework or application that makes use of these libraries. Examples of such projects are BlueBream, Grok and Zope 2.

If you want to use the Zope Toolkit KGS, you can use the buildout extends mechanism (replace 1.1 by the desired version):

```
[buildout]
extends = http://download.zope.org/zopetoolkit/index/1.1/ztk-versions.cfg
```

You can also copy the file locally or additionally extend the zopeapp-versions.cfg file from the same location.

Frameworks and applications have their own set of install instructions. You should follow these in most cases.

### Python versions

The ZTK 1.1 release series aim to support Python 2.5 up to Python 2.7. However, there is an issue with the *zope.proxy* package using Python 2.7 on 64-bit machines. This issue should be resolved with the upcoming release of Python 2.7.2. See also http://bugs.python.org/issue10360.

Some libraries included in the ZTK support Python 3.1 or later. But as a whole the ZTK supports no Python 3.x version yet.

### News

### Python versions

Python 2.4 is no longer supported by this version of the ZTK. Support for Python 2.7 has been added.

### ZODB 3.10

This ZTK version includes ZODB 3.10 instead of 3.9 as included in the 1.0 series. You can read more about the changes at http://pypi.python.org/pypi/ZODB3/3.10.0#change-history.

### zc.buildout 1.5

The buildout version has been updated to 1.5 from the former 1.4 series. This release requires some changes to recipes, so make sure to update all recipes to compatible versions or check their availability first. More detailed changes can be found at http://pypi.python.org/pypi/zc.buildout/1.5.2#change-history.

### zope.testing 3.10

In zope.testing 3.10 the *zope.testing.testrunner* package has been moved to a standalone distribution called *zope.testrunner*. You need to adjust your imports or use compatible versions of test runner recipes.

### Zope Toolkit 1.1.3

This document covers major changes in this release that can lead to backward-incompatibilities and explains what to look out for when updating.

### Introduction

The Zope Toolkit 1.1 release is the second feature release of the Zope Toolkit. The Zope Toolkit really is just a collection of libraries managed together by the Zope developers. We typically treat each library independently, so you would like to look at the CHANGES.txt in each library for updates. Here we note larger changes, especially ones that affect multiple libraries.

### Installation

The Zope Toolkit cannot be installed directly except as individual libraries (such as `zope.component`). To install it you typically would install a framework or application that makes use of these libraries. Examples of such projects are BlueBream, Grok and Zope 2.

If you want to use the Zope Toolkit KGS, you can use the buildout extends mechanism (replace 1.1 by the desired version):

```
[buildout]
extends = http://download.zope.org/zopetoolkit/index/1.1/ztk-versions.cfg
```

You can also copy the file locally or additionally extend the zopeapp-versions.cfg file from the same location.

Frameworks and applications have their own set of install instructions. You should follow these in most cases.

### Python versions

The ZTK 1.1 release series aim to support Python 2.5 up to Python 2.7. However, there is an issue with the *zope.proxy* package using Python 2.7 on 64-bit machines. This issue should be resolved with the upcoming release of Python 2.7.2. See also http://bugs.python.org/issue10360.

Some libraries included in the ZTK support Python 3.1 or later. But as a whole the ZTK supports no Python 3.x version yet.

#### News

#### Python versions

Python 2.4 is no longer supported by this version of the ZTK. Support for Python 2.7 has been added.

#### ZODB 3.10

This ZTK version includes ZODB 3.10 instead of 3.9 as included in the 1.0 series. You can read more about the changes at http://pypi.python.org/pypi/ZODB3/3.10.0#change-history.

#### zc.buildout 1.5

The buildout version has been updated to 1.5 from the former 1.4 series. This release requires some changes to recipes, so make sure to update all recipes to compatible versions or check their availability first. More detailed changes can be found at http://pypi.python.org/pypi/zc.buildout/1.5.2#change-history.

#### zope.testing 3.10

In zope.testing 3.10 the *zope.testing.testrunner* package has been moved to a standalone distribution called *zope.testrunner*. You need to adjust your imports or use compatible versions of test runner recipes.

#### Zope Toolkit 1.1.2

This document covers major changes in this release that can lead to backward-incompatibilities and explains what to look out for when updating.

#### Introduction

The Zope Toolkit 1.1 release is the second feature release of the Zope Toolkit. The Zope Toolkit really is just a collection of libraries managed together by the Zope developers. We typically treat each library independently, so you would like to look at the CHANGES.txt in each library for updates. Here we note larger changes, especially ones that affect multiple libraries.

#### Installation

The Zope Toolkit cannot be installed directly except as individual libraries (such as `zope.component`). To install it you typically would install a framework or application that makes use of these libraries. Examples of such projects are BlueBream, Grok and Zope 2.

If you want to use the Zope Toolkit KGS, you can use the buildout extends mechanism (replace 1.1 by the desired version):

```
[buildout]
extends = http://download.zope.org/zopetoolkit/index/1.1/ztk-versions.cfg
```

You can also copy the file locally or additionally extend the zopeapp-versions.cfg file from the same location.

Frameworks and applications have their own set of install instructions. You should follow these in most cases.

### Python versions

The ZTK 1.1 release series aim to support Python 2.5 up to Python 2.7. However, there is an issue with the *zope.proxy* package using Python 2.7 on 64-bit machines. This issue should be resolved with the upcoming release of Python 2.7.2. See also http://bugs.python.org/issue10360.

Some libraries included in the ZTK support Python 3.1 or later. But as a whole the ZTK supports no Python 3.x version yet.

### News

### Python versions

Python 2.4 is no longer supported by this version of the ZTK. Support for Python 2.7 has been added.

### ZODB 3.10

This ZTK version includes ZODB 3.10 instead of 3.9 as included in the 1.0 series. You can read more about the changes at http://pypi.python.org/pypi/ZODB3/3.10.0#change-history.

### zc.buildout 1.5

The buildout version has been updated to 1.5 from the former 1.4 series. This release requires some changes to recipes, so make sure to update all recipes to compatible versions or check their availability first. More detailed changes can be found at http://pypi.python.org/pypi/zc.buildout/1.5.2#change-history.

### zope.testing 3.10

In zope.testing 3.10 the *zope.testing.testrunner* package has been moved to a standalone distribution called *zope.testrunner*. You need to adjust your imports or use compatible versions of test runner recipes.

### Zope Toolkit 1.1.1

This document covers major changes in this release that can lead to backward-incompatibilities and explains what to look out for when updating.

### Introduction

The Zope Toolkit 1.1 release is the second feature release of the Zope Toolkit. The Zope Toolkit really is just a collection of libraries managed together by the Zope developers. We typically treat each library independently, so you would like to look at the CHANGES.txt in each library for updates. Here we note larger changes, especially ones that affect multiple libraries.

### Installation

The Zope Toolkit cannot be installed directly except as individual libraries (such as `zope.component`). To install it you typically would install a framework or application that makes use of these libraries. Examples of such projects are BlueBream, Grok and Zope 2.

If you want to use the Zope Toolkit KGS, you can use the buildout extends mechanism (replace 1.1 by the desired version):

```
[buildout]
extends = http://download.zope.org/zopetoolkit/index/1.1/ztk-versions.cfg
```

You can also copy the file locally or additionally extend the zopeapp-versions.cfg file from the same location.

Frameworks and applications have their own set of install instructions. You should follow these in most cases.

### Python versions

The ZTK 1.1 release series aim to support Python 2.5 up to Python 2.7. However, there is an issue with the *zope.proxy* package using Python 2.7 on 64-bit machines. This issue should be resolved with the upcoming release of Python 2.7.2. See also http://bugs.python.org/issue10360.

Some libraries included in the ZTK support Python 3.1 or later. But as a whole the ZTK supports no Python 3.x version yet.

### News

### Python versions

Python 2.4 is no longer supported by this version of the ZTK. Support for Python 2.7 has been added.

### ZODB 3.10

This ZTK version includes ZODB 3.10 instead of 3.9 as included in the 1.0 series. You can read more about the changes at http://pypi.python.org/pypi/ZODB3/3.10.0#change-history.

### zc.buildout 1.5

The buildout version has been updated to 1.5 from the former 1.4 series. This release requires some changes to recipes, so make sure to update all recipes to compatible versions or check their availability first. More detailed changes can be found at http://pypi.python.org/pypi/zc.buildout/1.5.2#change-history.

### zope.testing 3.10

In zope.testing 3.10 the *zope.testing.testrunner* package has been moved to a standalone distribution called *zope.testrunner*. You need to adjust your imports or use compatible versions of test runner recipes.

### Zope Toolkit 1.0

This document covers major changes in this release that can lead to backward-incompatibilities and explains what to look out for when updating.

### Introduction

The Zope Toolkit 1.0 release is the first release of the Zope Toolkit. The Zope Toolkit really is just a collection of libraries managed together by the Zope developers. We typically treat each library independently, so you would like to look at the CHANGES.txt in each library for updates. Here we note larger changes, especially ones that affect multiple libraries.

### Installation

The Zope Toolkit cannot be installed directly except as individual libraries (such as `zope.component`). To install it you typically would install a framework or application that makes use of these libraries. Examples of such projects are BlueBream, Grok and Zope 2.

If you want to use the Zope Toolkit KGS, you can use the buildout extends mechanism (replace 1.0c2 by the desired version):

```
[buildout]
extends = http://download.zope.org/zopetoolkit/index/1.0c2/ztk-versions.cfg
```

You can also copy the file locally or additionally extend the zopeapp-versions.cfg file from the same location.

Frameworks and applications have their own set of install instructions. You should follow these in most cases.

### News

The 1.0 release of the Zope Toolkit contains a number of refactorings that are aimed to clean up dependencies between pieces of code. Many packages in `zope.app` have had their code moved to existing or newly created packages in the `zope` namespace. These new packages do generally not contain user interface code (typically what's in `.browser`), and have much clearer dependency relationships as a result.

Backwards compatibility imports have been left in place so that your existing code should still work. In some cases you will have to explicitly add dependencies to a `zope.app.` to your code, as due to the cleanup they might not come in automatically anymore due to indirect dependencies; if you see an import error this is probably the case.

We recommend you update your existing code to import from the new packages if possible. The transition support and `zope.app.*` support is limited: the legacy support will be officially retired from the ZTK after January 1, 2011.

### Migration issues

### zope.app.keyreference -> zope.keyreference

This package was renamed to `zope.keyreference` and all its functionality was moved to the new one. The new package contains a little workaround for making old persistent keyreferences loadable without `zope.app.keyreference` installed, so the latter one is not needed at all anymore. Still review your code for any imports coming from `zope.app.keyreference` and modify it to use `zope.keyreference` instead.

### zope.app.intid -> zope.intid

The non-UI functionality of these packages was moved to `zope.intid` with backwards compatibility imports left in place. Review your imports from `zope.app.intid` to see whether they cannot come directly from `zope.intid` instead.

### zope.app.catalog -> zope.catalog

The non-UI functionality of these packages was moved to `zope.catalog`. Review your imports from `zope.app.catalog` to see whether they cannot come directly from `zope.catalog` instead.

### zope.app.container -> zope.container

The non-UI functionality of these packages was moved to `zope.container`. Review your imports from `zope.app.container` to see whether they cannot come directly from `zope.container` instead.

In addition, the exceptions used by `zope.container` were changed, so if your code catches them, you need to review it:

- The `DuplicationError` in `setitem` was changed to `KeyError`.
- The `UserError` in `NameChooser` was changed to `ValueError`.

### zope.app.component -> zope.security, zope.site

The implementation of the `<class>` ZCML directive moved from this package to `zope.security`. Packages that relied on `zope.app.component` to obtain this directive should declare a direct dependency on `zope.security`, and it may be possible to lose the dependency on `zope.app.component` altogether.

Non-UI site related functionality has been moved to the `zope.site` package. with backwards compatibility imports left in place. Review your imports from `zope.app.component` to see whether they cannot come directly from `zope.site` instead.

### zope.app.folder -> zope.site, zope.container

The implementation of the `zope.app.folder.Folder` class has moved to `zope.site.folder` instead, with backwards compatibility imports left in place. Review your imports from `zope.app.folder` to see whether they cannot come directly from `zope.site` instead. In addition, `Folder` is an `IContainer` implementation that also mixes in site management functionality. If such site management support is not necessary, in some cases your code does not need `Folder` but may be able to rely on a `Container` implementation from `zope.container` instead.

A base class with the implementation of the container-like behavior of `Folder` has moved to `zope.container` (and `zope.site` uses this for its implementation of `Folder`). This is not normally something you should need to retain backwards compatibility.

### zc.copy -> zope.copy, zope.copypastemove, zope.location

The pluggable object copying mechanism once developed in the `zc.copy` package was merged back into `zope.location`, `zope.copypastemove` and the new `zope.copy` package. The `zope.copy` package now provides a pluggable mechanism for copying objects from `zc.copy` and doesn't depend on anything but `zope.interface`. The `zope.copypastemove` uses the `copy` function from `zope.copy` in its `ObjectCopier`.

The `zope.location` now provides an `ICopyHook` adapter that implements conditional copy functionality based on object locations, that old `zope.location.pickling.CopyPersistent` used to provide. Note, that if you don't use ZCML configuration of `zope.location`, you may need to register `zope.location.pickling.LocationCopyHook` yourself.

The `zope.location.pickling.locationCopy` and `zope.location.pickling.CopyPersistent` are now deprecated in favor of `zope.copy` and were replaced by backward-compatibility imports. See `zope.copy` package documentation for information on how to use the new mechanism.

The new version of the `zc.copy` package now only contains backward-compatibility imports and is deprecated. `zope.copy` should be preferred for new developments.

### zope.app.security refactoring

The `zope.app.security` package was finally refactored into a few small parts with less dependencies and more clear purpose.

The implementation of the `<module>` ZCML directive moved from this package to `zope.security`. Packages that relied on `zope.app.security` to obtain this directive should declare a direct dependency on `zope.security`, and it may be possible to lose the dependency on `zope.app.security` altogether.

The `protectclass` module in this package has moved to `zope.security`, with backwards compatibility imports left in place. Review your imports from `zope.app.security` to see whether they cannot come directly from `zope.security` instead.

All interfaces (*IAuthentication*, *IUnauthenticatedPrincipal*, *ILoginPassword* and so on.) were moved into a new `zope.authentication` package, as well as several utility things, like *PrincipalSource* and *checkPrincipal* function. The new package has much less dependencies and defines an abstract contract for implementing authentication policies. While backward compatibility imports are left in place, it's strongly recommended to update your imports to the `zope.authentication`.

The *global principal registry* and its ZCML directives are moved into a new `zope.principalregistry` package with backward-compatibility imports left in place. If your application uses global principals, review your code and ZCML configuration to update it to the new place.

The *local permission* functionality was moved into a new `zope.app.localpermission` package. This functionality is a part of Through-The-Web development pattern that seems not to be used and supported much by Zope Toolkit and Application anymore, so it can be considered deprecated. However, it can serve as a great example of TTW-related component.

The *Permission vocabularies* and standard protections for Message objects and *__name__*, *__parent__* attributes as well as some common permissions, like *zope.View* and *zope.ManageContent* were merged into *zope.security*.

The adapters from `zope.publisher`'s *IHTTPCredentials* and *IFTPCredentials* to the *ILoginPassword* were moved into `zope.publisher`, thus making `zope.authentication` a dependency for `zope.publisher`.

The original `zope.app.security` package now only contains several deprecated or application-specific permission definitions, python module protections, that are only likely to be needed with deprecated Through-The-Web development pattern, and ZMI-related browser views (login.html, zope.app.form view for PrincipalSource and so on), as well as backward-compatibility imports. So, if you're not using TTW and/or standard ZMI browser views, you

probably should review update your imports to a new places and drop dependency on `zope.app.security` to reduce package dependencies count.

Other packages, that used `zope.app.security`, like `zope.securitypolicy` are either already adapted to the changes or will be adapted soon.

### zope.app.publisher refactoring

The `zope.app.publisher` package was also refactored into smaller parts with less dependencies and clearer purpose.

The browser resources mechanism (mostly used for serving static files and directories) was factored out to the new `zope.browserresource` package. It was also made more pluggable, so you can register specific resource classes for some file extensions, if you need special processing. One of the example is the new `zope.ptresource` package, where the PageTemplateResource was moved, another example is `z3c.zrtresource` package that was adapted to automatically use ZRT resource class for files with `.zrt` extensions.

Browser menu mechanism was moved into a new `zope.browsermenu` package with no further changes.

ZCML directives for easy creation of browser views (the `browser:page` directive and friends) was moved into a new small package, `zope.browserpage`. Also, the directives don't depend the menu mechanism now and will simply ignore "menu" and "title" arguments if `zope.browsermenu` package is not installed.

The `IModifiableBrowserLanguages` adapter was moved into `zope.publisher` along with several ZCML security declarations for `zope.publisher` classes that used to be in `zope.app.publisher`.

ZCML registrations for `IXMLRPCPublisher` adapter for containers was moved into the `zope.container`, because the actual adapters code were already in `zope.container` and registered there as `IBrowserPublisher` adapters. However, both adapters and their ZCML registrations will probably move elsewhere when we'll be refactoring `zope.container`.

Several parts are left in `zope.app.publisher` untouched:

- `Browser Skins` vocabulary.
- `date` field converter for `zope.publisher`'s form values conversion mechanism.
- `ManagementViewSelector` browser view (ZMI-related part).
- `xmlrpc:view` directive for publishing XML-RPC methods.

The latter, `xmlrpc:view` directive is generally useful, so it may be moved into a separate package in future, however there are no clear decision about how to move XML-RPC and FTP-related things currently.

### Password managers extracted from zope.app.authentication

The *IPasswordManager* interface and its implementations were extracted from `zope.app.authentication` into a new `zope.password` package to make them usable with other authentication systems, like `z3c.authenticator` or `zope.principalregistry` or any custom one.

It basically depends only on `zope.interface`, so it can be really useful even in non-Zope environments, like `Pylons`, for example.

The *Password Manager Names* vocabulary is also moved into `zope.password`, however, it's only useful with `zope.schema` and `zope.component`, so you need them installed to work with them. They're listed in the "vocabulary" extra requirement specification.

### ZODB 3.9 FileStorage native blob support

The FileStorage component of ZODB 3.9 used in Zope Toolkit 1.0 now supports blobs natively, so you don't need to use BlobStorage proxy for it anymore.

Thus, you can specify blob directory directly to FileStorage. If you use ZConfig, that means something like this:

```
<filestorage>
  path var/Data.fs
  blob-dir var/blobs
</filestorage>
```

instead of:

```
<blobstorage>
  blob-dir var/blobs
  <filestorage>
    path var/Data.fs
  </filestorage>
</blobstorage>
```

If you creating a storage from python, that means something like this:

```
storage = FileStorage('var/Data.fs', blob_dir='var/blobs')
```

instead of:

```
storage = BlobStorage('var/blobs', FileStorage('var/Data.fs'))
```

### zope.dublincore permission renaming

The `zope.app.dublincore.*` permissions have been renamed to `zope.dublincore.*`. Applications using these permissions have to fix up grants based on the old permissions.

### Zope Toolkit 1.0c3

This document covers major changes in this release that can lead to backward-incompatibilities and explains what to look out for when updating.

### Introduction

The Zope Toolkit 1.0 release is the first release of the Zope Toolkit. The Zope Toolkit really is just a collection of libraries managed together by the Zope developers. We typically treat each library independently, so you would like to look at the CHANGES.txt in each library for updates. Here we note larger changes, especially ones that affect multiple libraries.

### Installation

The Zope Toolkit cannot be installed directly except as individual libraries (such as `zope.component`). To install it you typically would install a framework or application that makes use of these libraries. Examples of such projects are BlueBream, Grok and Zope 2.

If you want to use the Zope Toolkit KGS, you can use the buildout extends mechanism (replace 1.0c2 by the desired version):

```
[buildout]
extends = http://download.zope.org/zopetoolkit/index/1.0c2/ztk-versions.cfg
```

You can also copy the file locally or additionally extend the zopeapp-versions.cfg file from the same location.

Frameworks and applications have their own set of install instructions. You should follow these in most cases.

### News

The 1.0 release of the Zope Toolkit contains a number of refactorings that are aimed to clean up dependencies between pieces of code. Many packages in `zope.app` have had their code moved to existing or newly created packages in the `zope` namespace. These new packages do generally not contain user interface code (typically what's in `.browser`), and have much clearer dependency relationships as a result.

Backwards compatibility imports have been left in place so that your existing code should still work. In some cases you will have to explicitly add dependencies to a `zope.app.` to your code, as due to the cleanup they might not come in automatically anymore due to indirect dependencies; if you see an import error this is probably the case.

We recommend you update your existing code to import from the new packages if possible. The transition support and `zope.app.*` support is limited: the legacy support will be officially retired from the ZTK after January 1, 2011.

### Migration issues

#### zope.app.keyreference -> zope.keyreference

This package was renamed to `zope.keyreference` and all its functionality was moved to the new one. The new package contains a little workaround for making old persistent keyreferences loadable without `zope.app.keyreference` installed, so the latter one is not needed at all anymore. Still review your code for any imports coming from `zope.app.keyreference` and modify it to use `zope.keyreference` instead.

#### zope.app.intid -> zope.intid

The non-UI functionality of these packages was moved to `zope.intid` with backwards compatibility imports left in place. Review your imports from `zope.app.intid` to see whether they cannot come directly from `zope.intid` instead.

#### zope.app.catalog -> zope.catalog

The non-UI functionality of these packages was moved to `zope.catalog`. Review your imports from `zope.app.catalog` to see whether they cannot come directly from `zope.catalog` instead.

#### zope.app.container -> zope.container

The non-UI functionality of these packages was moved to `zope.container`. Review your imports from `zope.app.container` to see whether they cannot come directly from `zope.container` instead.

In addition, the exceptions used by `zope.container` were changed, so if your code catches them, you need to review it:

- The `DuplicationError` in `setitem` was changed to `KeyError`.
- The `UserError` in `NameChooser` was changed to `ValueError`.

### zope.app.component -> zope.security, zope.site

The implementation of the `<class>` ZCML directive moved from this package to `zope.security`. Packages that relied on `zope.app.component` to obtain this directive should declare a direct dependency on `zope.security`, and it may be possible to lose the dependency on `zope.app.component` altogether.

Non-UI site related functionality has been moved to the `zope.site` package. with backwards compatibility imports left in place. Review your imports from `zope.app.component` to see whether they cannot come directly from `zope.site` instead.

### zope.app.folder -> zope.site, zope.container

The implementation of the `zope.app.folder.Folder` class has moved to `zope.site.folder` instead, with backwards compatibility imports left in place. Review your imports from `zope.app.folder` to see whether they cannot come directly from `zope.site` instead. In addition, `Folder` is an `IContainer` implementation that also mixes in site management functionality. If such site management support is not necessary, in some cases your code does not need `Folder` but may be able to rely on a `Container` implementation from `zope.container` instead.

A base class with the implementation of the container-like behavior of `Folder` has moved to `zope.container` (and `zope.site` uses this for its implementation of `Folder`). This is not normally something you should need to retain backwards compatibility.

### zc.copy -> zope.copy, zope.copypastemove, zope.location

The pluggable object copying mechanism once developed in the `zc.copy` package was merged back into `zope.location`, `zope.copypastemove` and the new `zope.copy` package. The `zope.copy` package now provides a pluggable mechanism for copying objects from `zc.copy` and doesn't depend on anything but `zope.interface`. The `zope.copypastemove` uses the `copy` function from `zope.copy` in its `ObjectCopier`.

The `zope.location` now provides an `ICopyHook` adapter that implements conditional copy functionality based on object locations, that old `zope.location.pickling.CopyPersistent` used to provide. Note, that if you don't use ZCML configuration of `zope.location`, you may need to register `zope.location.pickling.LocationCopyHook` yourself.

The `zope.location.pickling.locationCopy` and `zope.location.pickling.CopyPersistent` are now deprecated in favor of `zope.copy` and were replaced by backward-compatibility imports. See `zope.copy` package documentation for information on how to use the new mechanism.

The new version of the `zc.copy` package now only contains backward-compatibility imports and is deprecated. `zope.copy` should be preferred for new developments.

### zope.app.security refactoring

The `zope.app.security` package was finally refactored into a few small parts with less dependencies and more clear purpose.

The implementation of the `<module>` ZCML directive moved from this package to `zope.security`. Packages that relied on `zope.app.security` to obtain this directive should declare a direct dependency on `zope.security`, and it may be possible to lose the dependency on `zope.app.security` altogether.

The `protectclass` module in this package has moved to `zope.security`, with backwards compatibility imports left in place. Review your imports from `zope.app.security` to see whether they cannot come directly from `zope.security` instead.

All interfaces (*IAuthentication*, *IUnauthenticatedPrincipal*, *ILoginPassword* and so on.) were moved into a new `zope.authentication` package, as well as several utility things, like *PrincipalSource* and *checkPrincipal* function. The new package has much less dependencies and defines an abstract contract for implementing authentication policies. While backward compatibility imports are left in place, it's strongly recommended to update your imports to the `zope.authentication`.

The *global principal registry* and its ZCML directives are moved into a new `zope.principalregistry` package with backward-compatibility imports left in place. If your application uses global principals, review your code and ZCML configuration to update it to the new place.

The *local permission* functionality was moved into a new `zope.app.localpermission` package. This functionality is a part of Through-The-Web development pattern that seems not to be used and supported much by Zope Toolkit and Application anymore, so it can be considered deprecated. However, it can serve as a great example of TTW-related component.

The *Permission vocabularies* and standard protections for Message objects and *__name__*, *__parent__* attributes as well as some common permissions, like *zope.View* and *zope.ManageContent* were merged into *zope.security*.

The adapters from `zope.publisher`'s *IHTTPCredentials* and *IFTPCredentials* to the *ILoginPassword* were moved into `zope.publisher`, thus making `zope.authentication` a dependency for `zope.publisher`.

The original `zope.app.security` package now only contains several deprecated or application-specific permission definitions, python module protections, that are only likely to be needed with deprecated Through-The-Web development pattern, and ZMI-related browser views (login.html, zope.app.form view for PrincipalSource and so on), as well as backward-compatibility imports. So, if you're not using TTW and/or standard ZMI browser views, you probably should review update your imports to a new places and drop dependency on `zope.app.security` to reduce package dependencies count.

Other packages, that used `zope.app.security`, like `zope.securitypolicy` are either already adapted to the changes or will be adapted soon.

### zope.app.publisher refactoring

The `zope.app.publisher` package was also refactored into smaller parts with less dependencies and clearer purpose.

The browser resources mechanism (mostly used for serving static files and directories) was factored out to the new `zope.browserresource` package. It was also made more pluggable, so you can register specific resource classes for some file extensions, if you need special processing. One of the example is the new `zope.ptresource` package, where the PageTemplateResource was moved, another example is `z3c.zrtresource` package that was adapted to automatically use ZRT resource class for files with `.zrt` extensions.

Browser menu mechanism was moved into a new `zope.browsermenu` package with no further changes.

ZCML directives for easy creation of browser views (the `browser:page` directive and friends) was moved into a new small package, `zope.browserpage`. Also, the directives don't depend the menu mechanism now and will simply ignore "menu" and "title" arguments if `zope.browsermenu` package is not installed.

The `IModifiableBrowserLanguages` adapter was moved into `zope.publisher` along with several ZCML security declarations for `zope.publisher` classes that used to be in `zope.app.publisher`.

ZCML registrations for `IXMLRPCPublisher` adapter for containers was moved into the `zope.container`, because the actual adapters code were already in `zope.container` and registered there as `IBrowserPublisher` adapters. However, both adapters and their ZCML registrations will probably move elsewhere when we'll be refactoring `zope.container`.

Several parts are left in `zope.app.publisher` untouched:

- `Browser Skins` vocabulary.
- `date` field converter for `zope.publisher`'s form values conversion mechanism.
- `ManagementViewSelector` browser view (ZMI-related part).
- `xmlrpc:view` directive for publishing XML-RPC methods.

The latter, `xmlrpc:view` directive is generally useful, so it may be moved into a separate package in future, however there are no clear decision about how to move XML-RPC and FTP-related things currently.

### Password managers extracted from zope.app.authentication

The *IPasswordManager* interface and its implementations were extracted from `zope.app.authentication` into a new `zope.password` package to make them usable with other authentication systems, like `z3c.authenticator` or `zope.principalregistry` or any custom one.

It basically depends only on `zope.interface`, so it can be really useful even in non-Zope environments, like `Pylons`, for example.

The *Password Manager Names* vocabulary is also moved into `zope.password`, however, it's only useful with `zope.schema` and `zope.component`, so you need them installed to work with them. They're listed in the "vocabulary" extra requirement specification.

### ZODB 3.9 FileStorage native blob support

The FileStorage component of ZODB 3.9 used in Zope Toolkit 1.0 now supports blobs natively, so you don't need to use BlobStorage proxy for it anymore.

Thus, you can specify blob directory directly to FileStorage. If you use ZConfig, that means something like this:

```
<filestorage>
  path var/Data.fs
  blob-dir var/blobs
</filestorage>
```

instead of:

```
<blobstorage>
  blob-dir var/blobs
  <filestorage>
    path var/Data.fs
  </filestorage>
</blobstorage>
```

If you creating a storage from python, that means something like this:

```
storage = FileStorage('var/Data.fs', blob_dir='var/blobs')
```

instead of:

```
storage = BlobStorage('var/blobs', FileStorage('var/Data.fs'))
```

### zope.dublincore permission renaming

The `zope.app.dublincore.*` permissions have been renamed to `zope.dublincore.*`. Applications using these permissions have to fix up grants based on the old permissions.

## Zope Toolkit 1.0c2

This document covers major changes in this release that can lead to backward-incompatibilities and explains what to look out for when updating.

### Introduction

The Zope Toolkit 1.1 release is the second feature release of the Zope Toolkit. The Zope Toolkit really is just a collection of libraries managed together by the Zope developers. We typically treat each library independently, so you would like to look at the CHANGES.txt in each library for updates. Here we note larger changes, especially ones that affect multiple libraries.

### Installation

The Zope Toolkit cannot be installed directly except as individual libraries (such as `zope.component`). To install it you typically would install a framework or application that makes use of these libraries. Examples of such projects are BlueBream, Grok and Zope 2.

If you want to use the Zope Toolkit KGS, you can use the buildout extends mechanism (replace 1.1a1 by the desired version):

```
[buildout]
extends = http://download.zope.org/zopetoolkit/index/1.1a1/ztk-versions.cfg
```

You can also copy the file locally or additionally extend the zopeapp-versions.cfg file from the same location.

Frameworks and applications have their own set of install instructions. You should follow these in most cases.

## Zope Toolkit 1.0c1

This document covers major changes in this release that can lead to backward-incompatibilities and explains what to look out for when updating.

### Introduction

The Zope Toolkit 1.0 release is the first release of the Zope Toolkit. The Zope Toolkit really is just a collection of libraries managed together by the Zope developers. We typically treat each library independently, so you would like to look at the CHANGES.txt in each library for updates. Here we note larger changes, especially ones that affect multiple libraries.

### Installation

The Zope Toolkit cannot be installed directly except as individual libraries (such as `zope.component`). To install it you typically would install a framework or application that makes use of these libraries. Examples of such projects are BlueBream, Grok and Zope 2.

If you want to use the Zope Toolkit KGS, you can use the buildout extends mechanism (replace 1.0a2 by the desired version):

```
[buildout]
extends = http://download.zope.org/zopetoolkit/index/1.0a2/ztk-versions.cfg
```

You can also copy the file locally or additionally extend the zopeapp-versions.cfg file from the same location.

Frameworks and applications have their own set of install instructions. You should follow these in most cases.

### News

The 1.0 release of the Zope Toolkit contains a number of refactorings that are aimed to clean up dependencies between pieces of code. Many packages in `zope.app` have had their code moved to existing or newly created packages in the `zope` namespace. These new packages do generally not contain user interface code (typically what's in `.browser`), and have much clearer dependency relationships as a result.

Backwards compatibility imports have been left in place so that your existing code should still work. In some cases you will have to explicitly add dependencies to a `zope.app.` to your code, as due to the cleanup they might not come in automatically anymore due to indirect dependencies; if you see an import error this is probably the case.

We recommend you update your existing code to import from the new packages if possible. The transition support and `zope.app.*` support is limited: the legacy support will be officially retired from the ZTK after January 1, 2011.

### Migration issues

#### zope.app.keyreference -> zope.keyreference

This package was renamed to `zope.keyreference` and all its functionality was moved to the new one. The new package contains a little workaround for making old persistent keyreferences loadable without `zope.app.keyreference` installed, so the latter one is not needed at all anymore. Still review your code for any imports coming from `zope.app.keyreference` and modify it to use `zope.keyreference` instead.

#### zope.app.intid -> zope.intid

The non-UI functionality of these packages was moved to `zope.intid` with backwards compatibility imports left in place. Review your imports from `zope.app.intid` to see whether they cannot come directly from `zope.intid` instead.

#### zope.app.catalog -> zope.catalog

The non-UI functionality of these packages was moved to `zope.catalog`. Review your imports from `zope.app.catalog` to see whether they cannot come directly from `zope.catalog` instead.

#### zope.app.container -> zope.container

The non-UI functionality of these packages was moved to `zope.container`. Review your imports from `zope.app.container` to see whether they cannot come directly from `zope.container` instead.

In addition, the exceptions used by `zope.container` were changed, so if your code catches them, you need to review it:

- The `DuplicationError` in `setitem` was changed to `KeyError`.
- The `UserError` in `NameChooser` was changed to `ValueError`.

### zope.app.component -> zope.security, zope.site

The implementation of the `<class>` ZCML directive moved from this package to `zope.security`. Packages that relied on `zope.app.component` to obtain this directive should declare a direct dependency on `zope.security`, and it may be possible to lose the dependency on `zope.app.component` altogether.

Non-UI site related functionality has been moved to the `zope.site` package. with backwards compatibility imports left in place. Review your imports from `zope.app.component` to see whether they cannot come directly from `zope.site` instead.

### zope.app.folder -> zope.site, zope.container

The implementation of the `zope.app.folder.Folder` class has moved to `zope.site.folder` instead, with backwards compatibility imports left in place. Review your imports from `zope.app.folder` to see whether they cannot come directly from `zope.site` instead. In addition, `Folder` is an `IContainer` implementation that also mixes in site management functionality. If such site management support is not necessary, in some cases your code does not need `Folder` but may be able to rely on a `Container` implementation from `zope.container` instead.

A base class with the implementation of the container-like behavior of `Folder` has moved to `zope.container` (and `zope.site` uses this for its implementation of `Folder`). This is not normally something you should need to retain backwards compatibility.

### zc.copy -> zope.copy, zope.copypastemove, zope.location

The pluggable object copying mechanism once developed in the `zc.copy` package was merged back into `zope.location`, `zope.copypastemove` and the new `zope.copy` package. The `zope.copy` package now provides a pluggable mechanism for copying objects from `zc.copy` and doesn't depend on anything but `zope.interface`. The `zope.copypastemove` uses the `copy` function from `zope.copy` in its `ObjectCopier`.

The `zope.location` now provides an `ICopyHook` adapter that implements conditional copy functionality based on object locations, that old `zope.location.pickling.CopyPersistent` used to provide. Note, that if you don't use ZCML configuration of `zope.location`, you may need to register `zope.location.pickling.LocationCopyHook` yourself.

The `zope.location.pickling.locationCopy` and `zope.location.pickling.CopyPersistent` are now deprecated in favor of `zope.copy` and were replaced by backward-compatibility imports. See `zope.copy` package documentation for information on how to use the new mechanism.

The new version of the `zc.copy` package now only contains backward-compatibility imports and is deprecated. `zope.copy` should be preferred for new developments.

### zope.app.security refactoring

The `zope.app.security` package was finally refactored into a few small parts with less dependencies and more clear purpose.

The implementation of the `<module>` ZCML directive moved from this package to `zope.security`. Packages that relied on `zope.app.security` to obtain this directive should declare a direct dependency on `zope.security`, and it may be possible to lose the dependency on `zope.app.security` altogether.

The `protectclass` module in this package has moved to `zope.security`, with backwards compatibility imports left in place. Review your imports from `zope.app.security` to see whether they cannot come directly from `zope.security` instead.

All interfaces (*IAuthentication*, *IUnauthenticatedPrincipal*, *ILoginPassword* and so on.) were moved into a new `zope.authentication` package, as well as several utility things, like *PrincipalSource* and *checkPrincipal* function. The new package has much less dependencies and defines an abstract contract for implementing authentication policies. While backward compatibility imports are left in place, it's strongly recommended to update your imports to the `zope.authentication`.

The *global principal registry* and its ZCML directives are moved into a new `zope.principalregistry` package with backward-compatibility imports left in place. If your application uses global principals, review your code and ZCML configuration to update it to the new place.

The *local permission* functionality was moved into a new `zope.app.localpermission` package. This functionality is a part of Through-The-Web development pattern that seems not to be used and supported much by Zope Toolkit and Application anymore, so it can be considered deprecated. However, it can serve as a great example of TTW-related component.

The *Permission vocabularies* and standard protections for Message objects and *__name__*, *__parent__* attributes as well as some common permissions, like *zope.View* and *zope.ManageContent* were merged into *zope.security*.

The adapters from `zope.publisher`'s *IHTTPCredentials* and *IFTPCredentials* to the *ILoginPassword* were moved into `zope.publisher`, thus making `zope.authentication` a dependency for `zope.publisher`.

The original `zope.app.security` package now only contains several deprecated or application-specific permission definitions, python module protections, that are only likely to be needed with deprecated Through-The-Web development pattern, and ZMI-related browser views (login.html, zope.app.form view for PrincipalSource and so on), as well as backward-compatibility imports. So, if you're not using TTW and/or standard ZMI browser views, you probably should review update your imports to a new places and drop dependency on `zope.app.security` to reduce package dependencies count.

Other packages, that used `zope.app.security`, like `zope.securitypolicy` are either already adapted to the changes or will be adapted soon.

### zope.app.publisher refactoring

The `zope.app.publisher` package was also refactored into smaller parts with less dependencies and clearer purpose.

The browser resources mechanism (mostly used for serving static files and directories) was factored out to the new `zope.browserresource` package. It was also made more pluggable, so you can register specific resource classes for some file extensions, if you need special processing. One of the example is the new `zope.ptresource` package, where the PageTemplateResource was moved, another example is `z3c.zrtresource` package that was adapted to automatically use ZRT resource class for files with `.zrt` extensions.

Browser menu mechanism was moved into a new `zope.browsermenu` package with no further changes.

ZCML directives for easy creation of browser views (the `browser:page` directive and friends) was moved into a new small package, `zope.browserpage`. Also, the directives don't depend the menu mechanism now and will simply ignore "menu" and "title" arguments if `zope.browsermenu` package is not installed.

The `IModifiableBrowserLanguages` adapter was moved into `zope.publisher` along with several ZCML security declarations for `zope.publisher` classes that used to be in `zope.app.publisher`.

ZCML registrations for `IXMLRPCPublisher` adapter for containers was moved into the `zope.container`, because the actual adapters code were already in `zope.container` and registered there as `IBrowserPublisher` adapters. However, both adapters and their ZCML registrations will probably move elsewhere when we'll be refactoring `zope.container`.

Several parts are left in `zope.app.publisher` untouched:

- `Browser Skins` vocabulary.
- `date` field converter for `zope.publisher`'s form values conversion mechanism.
- `ManagementViewSelector` browser view (ZMI-related part).
- `xmlrpc:view` directive for publishing XML-RPC methods.

The latter, `xmlrpc:view` directive is generally useful, so it may be moved into a separate package in future, however there are no clear decision about how to move XML-RPC and FTP-related things currently.

### Password managers extracted from zope.app.authentication

The *IPasswordManager* interface and its implementations were extracted from `zope.app.authentication` into a new `zope.password` package to make them usable with other authentication systems, like `z3c.authenticator` or `zope.principalregistry` or any custom one.

It basically depends only on `zope.interface`, so it can be really useful even in non-Zope environments, like `Pylons`, for example.

The *Password Manager Names* vocabulary is also moved into `zope.password`, however, it's only useful with `zope.schema` and `zope.component`, so you need them installed to work with them. They're listed in the "vocabulary" extra requirement specification.

### ZODB 3.9 FileStorage native blob support

The FileStorage component of ZODB 3.9 used in Zope Toolkit 1.0 now supports blobs natively, so you don't need to use BlobStorage proxy for it anymore.

Thus, you can specify blob directory directly to FileStorage. If you use ZConfig, that means something like this:

```
<filestorage>
  path var/Data.fs
  blob-dir var/blobs
</filestorage>
```

instead of:

```
<blobstorage>
  blob-dir var/blobs
  <filestorage>
    path var/Data.fs
  </filestorage>
</blobstorage>
```

If you creating a storage from python, that means something like this:

```
storage = FileStorage('var/Data.fs', blob_dir='var/blobs')
```

instead of:

```
storage = BlobStorage('var/blobs', FileStorage('var/Data.fs'))
```

### zope.dublincore permission renaming

The `zope.app.dublincore.*` permissions have been renamed to `zope.dublincore.*`. Applications using these permissions have to fix up grants based on the old permissions.

### Zope Toolkit 1.0a3

This document covers major changes in this release that can lead to backward-incompatibilities and explains what to look out for when updating.

### Introduction

The Zope Toolkit 1.0 release is the first release of the Zope Toolkit. The Zope Toolkit really is just a collection of libraries managed together by the Zope developers. We typically treat each library independently, so you would like to look at the CHANGES.txt in each library for updates. Here we note larger changes, especially ones that affect multiple libraries.

### Installation

The Zope Toolkit cannot be installed directly except as individual libraries (such as `zope.component`). To install it you typically would install a framework or application that makes use of these libraries. Examples of such projects are BlueBream, Grok and Zope 2.

If you want to use the Zope Toolkit KGS, you can use the buildout extends mechanism (replace 1.0a2 by the desired version):

```
[buildout]
extends = http://download.zope.org/zopetoolkit/index/1.0a2/ztk-versions.cfg
```

You can also copy the file locally or additionally extend the zopeapp-versions.cfg file from the same location.

Frameworks and applications have their own set of install instructions. You should follow these in most cases.

### News

The 1.0 release of the Zope Toolkit contains a number of refactorings that are aimed to clean up dependencies between pieces of code. Many packages in `zope.app` have had their code moved to existing or newly created packages in the `zope` namespace. These new packages do generally not contain user interface code (typically what's in `.browser`), and have much clearer dependency relationships as a result.

Backwards compatibility imports have been left in place so that your existing code should still work. In some cases you will have to explicitly add dependencies to a `zope.app.` to your code, as due to the cleanup they might not come in automatically anymore due to indirect dependencies; if you see an import error this is probably the case.

We recommend you update your existing code to import from the new packages if possible. The transition support and `zope.app.*` support is limited: the legacy support will be officially retired from the ZTK after January 1, 2011.

### Migration issues

### zope.app.keyreference -> zope.keyreference

This package was renamed to `zope.keyreference` and all its functionality was moved to the new one. The new package contains a little workaround for making old persistent keyreferences loadable without `zope.app.keyreference` installed, so the latter one is not needed at all anymore. Still review your code for any imports coming from `zope.app.keyreference` and modify it to use `zope.keyreference` instead.

### zope.app.intid -> zope.intid

The non-UI functionality of these packages was moved to `zope.intid` with backwards compatibility imports left in place. Review your imports from `zope.app.intid` to see whether they cannot come directly from `zope.intid` instead.

### zope.app.catalog -> zope.catalog

The non-UI functionality of these packages was moved to `zope.catalog`. Review your imports from `zope.app.catalog` to see whether they cannot come directly from `zope.catalog` instead.

### zope.app.container -> zope.container

The non-UI functionality of these packages was moved to `zope.container`. Review your imports from `zope.app.container` to see whether they cannot come directly from `zope.container` instead.

In addition, the exceptions used by `zope.container` were changed, so if your code catches them, you need to review it:

- The `DuplicationError` in `setitem` was changed to `KeyError`.
- The `UserError` in `NameChooser` was changed to `ValueError`.

### zope.app.component -> zope.security, zope.site

The implementation of the `<class>` ZCML directive moved from this package to `zope.security`. Packages that relied on `zope.app.component` to obtain this directive should declare a direct dependency on `zope.security`, and it may be possible to lose the dependency on `zope.app.component` altogether.

Non-UI site related functionality has been moved to the `zope.site` package. with backwards compatibility imports left in place. Review your imports from `zope.app.component` to see whether they cannot come directly from `zope.site` instead.

### zope.app.folder -> zope.site, zope.container

The implementation of the `zope.app.folder.Folder` class has moved to `zope.site.folder` instead, with backwards compatibility imports left in place. Review your imports from `zope.app.folder` to see whether they cannot come directly from `zope.site` instead. In addition, `Folder` is an `IContainer` implementation that also mixes in site management functionality. If such site management support is not necessary, in some cases your code does not need `Folder` but may be able to rely on a `Container` implementation from `zope.container` instead.

A base class with the implementation of the container-like behavior of `Folder` has moved to `zope.container` (and `zope.site` uses this for its implementation of `Folder`). This is not normally something you should need to retain backwards compatibility.

### zc.copy -> zope.copy, zope.copypastemove, zope.location

The pluggable object copying mechanism once developed in the `zc.copy` package was merged back into `zope.location`, `zope.copypastemove` and the new `zope.copy` package. The `zope.copy` package now provides a pluggable mechanism for copying objects from `zc.copy` and doesn't depend on anything but `zope.interface`. The `zope.copypastemove` uses the `copy` function from `zope.copy` in its `ObjectCopier`.

The `zope.location` now provides an `ICopyHook` adapter that implements conditional copy functionality based on object locations, that old `zope.location.pickling.CopyPersistent` used to provide. Note, that if you don't use ZCML configuration of `zope.location`, you may need to register `zope.location.pickling.LocationCopyHook` yourself.

The `zope.location.pickling.locationCopy` and `zope.location.pickling.CopyPersistent` are now deprecated in favor of `zope.copy` and were replaced by backward-compatibility imports. See `zope.copy` package documentation for information on how to use the new mechanism.

The new version of the `zc.copy` package now only contains backward-compatibility imports and is deprecated. `zope.copy` should be preferred for new developments.

### zope.app.security refactoring

The `zope.app.security` package was finally refactored into a few small parts with less dependencies and more clear purpose.

The implementation of the `<module>` ZCML directive moved from this package to `zope.security`. Packages that relied on `zope.app.security` to obtain this directive should declare a direct dependency on `zope.security`, and it may be possible to lose the dependency on `zope.app.security` altogether.

The `protectclass` module in this package has moved to `zope.security`, with backwards compatibility imports left in place. Review your imports from `zope.app.security` to see whether they cannot come directly from `zope.security` instead.

All interfaces (*IAuthentication*, *IUnauthenticatedPrincipal*, *ILoginPassword* and so on.) were moved into a new `zope.authentication` package, as well as several utility things, like *PrincipalSource* and *checkPrincipal* function. The new package has much less dependencies and defines an abstract contract for implementing authentication policies. While backward compatibility imports are left in place, it's strongly recommended to update your imports to the `zope.authentication`.

The *global principal registry* and its ZCML directives are moved into a new `zope.principalregistry` package with backward-compatibility imports left in place. If your application uses global principals, review your code and ZCML configuration to update it to the new place.

The *local permission* functionality was moved into a new `zope.app.localpermission` package. This functionality is a part of Through-The-Web development pattern that seems not to be used and supported much by Zope Toolkit and Application anymore, so it can be considered deprecated. However, it can serve as a great example of TTW-related component.

The *Permission vocabularies* and standard protections for Message objects and *__name__*, *__parent__* attributes as well as some common permissions, like *zope.View* and *zope.ManageContent* were merged into *zope.security*.

The adapters from `zope.publisher`'s *IHTTPCredentials* and *IFTPCredentials* to the *ILoginPassword* were moved into `zope.publisher`, thus making `zope.authentication` a dependency for `zope.publisher`.

The original `zope.app.security` package now only contains several deprecated or application-specific permission definitions, python module protections, that are only likely to be needed with deprecated Through-The-Web development pattern, and ZMI-related browser views (login.html, zope.app.form view for PrincipalSource and so on), as well as backward-compatibility imports. So, if you're not using TTW and/or standard ZMI browser views, you

probably should review update your imports to a new places and drop dependency on `zope.app.security` to reduce package dependencies count.

Other packages, that used `zope.app.security`, like `zope.securitypolicy` are either already adapted to the changes or will be adapted soon.

### zope.app.publisher refactoring

The `zope.app.publisher` package was also refactored into smaller parts with less dependencies and clearer purpose.

The browser resources mechanism (mostly used for serving static files and directories) was factored out to the new `zope.browserresource` package. It was also made more pluggable, so you can register specific resource classes for some file extensions, if you need special processing. One of the example is the new `zope.ptresource` package, where the PageTemplateResource was moved, another example is `z3c.zrtresource` package that was adapted to automatically use ZRT resource class for files with `.zrt` extensions.

Browser menu mechanism was moved into a new `zope.browsermenu` package with no further changes.

ZCML directives for easy creation of browser views (the `browser:page` directive and friends) was moved into a new small package, `zope.browserpage`. Also, the directives don't depend the menu mechanism now and will simply ignore "menu" and "title" arguments if `zope.browsermenu` package is not installed.

The `IModifiableBrowserLanguages` adapter was moved into `zope.publisher` along with several ZCML security declarations for `zope.publisher` classes that used to be in `zope.app.publisher`.

ZCML registrations for `IXMLRPCPublisher` adapter for containers was moved into the `zope.container`, because the actual adapters code were already in `zope.container` and registered there as `IBrowserPublisher` adapters. However, both adapters and their ZCML registrations will probably move elsewhere when we'll be refactoring `zope.container`.

Several parts are left in `zope.app.publisher` untouched:

- `Browser Skins` vocabulary.
- `date` field converter for `zope.publisher`'s form values conversion mechanism.
- `ManagementViewSelector` browser view (ZMI-related part).
- `xmlrpc:view` directive for publishing XML-RPC methods.

The latter, `xmlrpc:view` directive is generally useful, so it may be moved into a separate package in future, however there are no clear decision about how to move XML-RPC and FTP-related things currently.

### Password managers extracted from zope.app.authentication

The *IPasswordManager* interface and its implementations were extracted from `zope.app.authentication` into a new `zope.password` package to make them usable with other authentication systems, like `z3c.authenticator` or `zope.principalregistry` or any custom one.

It basically depends only on `zope.interface`, so it can be really useful even in non-Zope environments, like `Pylons`, for example.

The *Password Manager Names* vocabulary is also moved into `zope.password`, however, it's only useful with `zope.schema` and `zope.component`, so you need them installed to work with them. They're listed in the "vocabulary" extra requirement specification.

### ZODB 3.9 FileStorage native blob support

The FileStorage component of ZODB 3.9 used in Zope Toolkit 1.0 now supports blobs natively, so you don't need to use BlobStorage proxy for it anymore.

Thus, you can specify blob directory directly to FileStorage. If you use ZConfig, that means something like this:

```
<filestorage>
  path var/Data.fs
  blob-dir var/blobs
</filestorage>
```

instead of:

```
<blobstorage>
  blob-dir var/blobs
  <filestorage>
    path var/Data.fs
  </filestorage>
</blobstorage>
```

If you creating a storage from python, that means something like this:

```
storage = FileStorage('var/Data.fs', blob_dir='var/blobs')
```

instead of:

```
storage = BlobStorage('var/blobs', FileStorage('var/Data.fs'))
```

### zope.dublincore permission renaming

The `zope.app.dublincore.*` permissions have been renamed to `zope.dublincore.*`. Applications using these permissions have to fix up grants based on the old permissions.

### Zope Toolkit 1.0a2

This document covers major changes in this release that can lead to backward-incompatibilities and explains what to look out for when updating.

### Introduction

The Zope Toolkit 1.0 release is the first release of the Zope Toolkit. The Zope Toolkit really is just a collection of libraries managed together by the Zope developers. We typically treat each library independently, so you would like to look at the CHANGES.txt in each library for updates. Here we note larger changes, especially ones that affect multiple libraries.

### Installation

The Zope Toolkit cannot be installed directly except as individual libraries (such as `zope.component`). To install it you typically would install a framework or application that makes use of these libraries. Examples of such projects are BlueBream, Grok and Zope 2.

If you want to use the Zope Toolkit KGS, you can use the buildout extends mechanism (replace 1.0a2 by the desired version):

```
[buildout]
extends = http://download.zope.org/zopetoolkit/index/1.0a2/ztk-versions.cfg
```

You can also copy the file locally or additionally extend the zopeapp-versions.cfg file from the same location.

Frameworks and applications have their own set of install instructions. You should follow these in most cases.

### News

The 1.0 release of the Zope Toolkit contains a number of refactorings that are aimed to clean up dependencies between pieces of code. Many packages in `zope.app` have had their code moved to existing or newly created packages in the `zope` namespace. These new packages do generally not contain user interface code (typically what's in `.browser`), and have much clearer dependency relationships as a result.

Backwards compatibility imports have been left in place so that your existing code should still work. In some cases you will have to explicitly add dependencies to a `zope.app.` to your code, as due to the cleanup they might not come in automatically anymore due to indirect dependencies; if you see an import error this is probably the case.

We recommend you update your existing code to import from the new packages if possible. The transition support and `zope.app.*` support is limited: the legacy support will be officially retired from the ZTK after January 1, 2011.

### Migration issues

#### zope.app.keyreference -> zope.keyreference

This package was renamed to `zope.keyreference` and all its functionality was moved to the new one. The new package contains a little workaround for making old persistent keyreferences loadable without `zope.app.keyreference` installed, so the latter one is not needed at all anymore. Still review your code for any imports coming from `zope.app.keyreference` and modify it to use `zope.keyreference` instead.

#### zope.app.intid -> zope.intid

The non-UI functionality of these packages was moved to `zope.intid` with backwards compatibility imports left in place. Review your imports from `zope.app.intid` to see whether they cannot come directly from `zope.intid` instead.

#### zope.app.catalog -> zope.catalog

The non-UI functionality of these packages was moved to `zope.catalog`. Review your imports from `zope.app.catalog` to see whether they cannot come directly from `zope.catalog` instead.

#### zope.app.container -> zope.container

The non-UI functionality of these packages was moved to `zope.container`. Review your imports from `zope.app.container` to see whether they cannot come directly from `zope.container` instead.

In addition, the exceptions used by `zope.container` were changed, so if your code catches them, you need to review it:

- The `DuplicationError` in `setitem` was changed to `KeyError`.
- The `UserError` in `NameChooser` was changed to `ValueError`.

### zope.app.component -> zope.security, zope.site

The implementation of the `<class>` ZCML directive moved from this package to `zope.security`. Packages that relied on `zope.app.component` to obtain this directive should declare a direct dependency on `zope.security`, and it may be possible to lose the dependency on `zope.app.component` altogether.

Non-UI site related functionality has been moved to the `zope.site` package. with backwards compatibility imports left in place. Review your imports from `zope.app.component` to see whether they cannot come directly from `zope.site` instead.

### zope.app.folder -> zope.site, zope.container

The implementation of the `zope.app.folder.Folder` class has moved to `zope.site.folder` instead, with backwards compatibility imports left in place. Review your imports from `zope.app.folder` to see whether they cannot come directly from `zope.site` instead. In addition, `Folder` is an `IContainer` implementation that also mixes in site management functionality. If such site management support is not necessary, in some cases your code does not need `Folder` but may be able to rely on a `Container` implementation from `zope.container` instead.

A base class with the implementation of the container-like behavior of `Folder` has moved to `zope.container` (and `zope.site` uses this for its implementation of `Folder`). This is not normally something you should need to retain backwards compatibility.

### zc.copy -> zope.copy, zope.copypastemove, zope.location

The pluggable object copying mechanism once developed in the `zc.copy` package was merged back into `zope.location`, `zope.copypastemove` and the new `zope.copy` package. The `zope.copy` package now provides a pluggable mechanism for copying objects from `zc.copy` and doesn't depend on anything but `zope.interface`. The `zope.copypastemove` uses the `copy` function from `zope.copy` in its `ObjectCopier`.

The `zope.location` now provides an `ICopyHook` adapter that implements conditional copy functionality based on object locations, that old `zope.location.pickling.CopyPersistent` used to provide. Note, that if you don't use ZCML configuration of `zope.location`, you may need to register `zope.location.pickling.LocationCopyHook` yourself.

The `zope.location.pickling.locationCopy` and `zope.location.pickling.CopyPersistent` are now deprecated in favor of `zope.copy` and were replaced by backward-compatibility imports. See `zope.copy` package documentation for information on how to use the new mechanism.

The new version of the `zc.copy` package now only contains backward-compatibility imports and is deprecated. `zope.copy` should be preferred for new developments.

### zope.app.security refactoring

The `zope.app.security` package was finally refactored into a few small parts with less dependencies and more clear purpose.

The implementation of the `<module>` ZCML directive moved from this package to `zope.security`. Packages that relied on `zope.app.security` to obtain this directive should declare a direct dependency on `zope.security`, and it may be possible to lose the dependency on `zope.app.security` altogether.

The `protectclass` module in this package has moved to `zope.security`, with backwards compatibility imports left in place. Review your imports from `zope.app.security` to see whether they cannot come directly from `zope.security` instead.

All interfaces (*IAuthentication*, *IUnauthenticatedPrincipal*, *ILoginPassword* and so on.) were moved into a new `zope.authentication` package, as well as several utility things, like *PrincipalSource* and *checkPrincipal* function. The new package has much less dependencies and defines an abstract contract for implementing authentication policies. While backward compatibility imports are left in place, it's strongly recommended to update your imports to the `zope.authentication`.

The *global principal registry* and its ZCML directives are moved into a new `zope.principalregistry` package with backward-compatibility imports left in place. If your application uses global principals, review your code and ZCML configuration to update it to the new place.

The *local permission* functionality was moved into a new `zope.app.localpermission` package. This functionality is a part of Through-The-Web development pattern that seems not to be used and supported much by Zope Toolkit and Application anymore, so it can be considered deprecated. However, it can serve as a great example of TTW-related component.

The *Permission vocabularies* and standard protections for Message objects and *__name__*, *__parent__* attributes as well as some common permissions, like *zope.View* and *zope.ManageContent* were merged into *zope.security*.

The adapters from `zope.publisher`'s *IHTTPCredentials* and *IFTPCredentials* to the *ILoginPassword* were moved into `zope.publisher`, thus making `zope.authentication` a dependency for `zope.publisher`.

The original `zope.app.security` package now only contains several deprecated or application-specific permission definitions, python module protections, that are only likely to be needed with deprecated Through-The-Web development pattern, and ZMI-related browser views (login.html, zope.app.form view for PrincipalSource and so on), as well as backward-compatibility imports. So, if you're not using TTW and/or standard ZMI browser views, you probably should review update your imports to a new places and drop dependency on `zope.app.security` to reduce package dependencies count.

Other packages, that used `zope.app.security`, like `zope.securitypolicy` are either already adapted to the changes or will be adapted soon.

### zope.app.publisher refactoring

The `zope.app.publisher` package was also refactored into smaller parts with less dependencies and clearer purpose.

The browser resources mechanism (mostly used for serving static files and directories) was factored out to the new `zope.browserresource` package. It was also made more pluggable, so you can register specific resource classes for some file extensions, if you need special processing. One of the example is the new `zope.ptresource` package, where the PageTemplateResource was moved, another example is `z3c.zrtresource` package that was adapted to automatically use ZRT resource class for files with `.zrt` extensions.

Browser menu mechanism was moved into a new `zope.browsermenu` package with no further changes.

ZCML directives for easy creation of browser views (the `browser:page` directive and friends) was moved into a new small package, `zope.browserpage`. Also, the directives don't depend the menu mechanism now and will simply ignore "menu" and "title" arguments if `zope.browsermenu` package is not installed.

The `IModifiableBrowserLanguages` adapter was moved into `zope.publisher` along with several ZCML security declarations for `zope.publisher` classes that used to be in `zope.app.publisher`.

ZCML registrations for `IXMLRPCPublisher` adapter for containers was moved into the `zope.container`, because the actual adapters code were already in `zope.container` and registered there as `IBrowserPublisher` adapters. However, both adapters and their ZCML registrations will probably move elsewhere when we'll be refactoring `zope.container`.

Several parts are left in `zope.app.publisher` untouched:

- `Browser Skins` vocabulary.
- `date` field converter for `zope.publisher`'s form values conversion mechanism.
- `ManagementViewSelector` browser view (ZMI-related part).
- `xmlrpc:view` directive for publishing XML-RPC methods.

The latter, `xmlrpc:view` directive is generally useful, so it may be moved into a separate package in future, however there are no clear decision about how to move XML-RPC and FTP-related things currently.

### Password managers extracted from zope.app.authentication

The *IPasswordManager* interface and its implementations were extracted from `zope.app.authentication` into a new `zope.password` package to make them usable with other authentication systems, like `z3c.authenticator` or `zope.principalregistry` or any custom one.

It basically depends only on `zope.interface`, so it can be really useful even in non-Zope environments, like `Pylons`, for example.

The *Password Manager Names* vocabulary is also moved into `zope.password`, however, it's only useful with `zope.schema` and `zope.component`, so you need them installed to work with them. They're listed in the "vocabulary" extra requirement specification.

### ZODB 3.9 FileStorage native blob support

The FileStorage component of ZODB 3.9 used in Zope Toolkit 1.0 now supports blobs natively, so you don't need to use BlobStorage proxy for it anymore.

Thus, you can specify blob directory directly to FileStorage. If you use ZConfig, that means something like this:

```
<filestorage>
  path var/Data.fs
  blob-dir var/blobs
</filestorage>
```

instead of:

```
<blobstorage>
  blob-dir var/blobs
  <filestorage>
    path var/Data.fs
  </filestorage>
</blobstorage>
```

If you creating a storage from python, that means something like this:

```
storage = FileStorage('var/Data.fs', blob_dir='var/blobs')
```

instead of:

```
storage = BlobStorage('var/blobs', FileStorage('var/Data.fs'))
```

### Zope Toolkit 1.0a1

This document covers major changes in this release that can lead to backward-incompatibilities and explains what to look out for when updating.

### Introduction

The Zope Toolkit 1.0 release is the first release of the Zope Toolkit. The Zope Toolkit really is just a collection of libraries managed together by the Zope developers. We typically treat each library independently, so you would like to look at the CHANGES.txt in each library for updates. Here we note larger changes, especially ones that affect multiple libraries.

### Installation

The Zope Toolkit cannot be installed directly except as individual libraries (such as `zope.component`). To install it you typically would install a framework or application that makes use of these libraries. Examples of such projects are BlueBream, Grok and Zope 2.

If you want to use the Zope Toolkit KGS, you can use the buildout extends mechanism (replace 1.0a1 by the desired version):

```
[buildout]
extends = http://download.zope.org/zopetoolkit/index/1.0a1/ztk-versions.cfg
```

You can also copy the file locally or additionally extend the zopeapp-versions.cfg file from the same location.

Frameworks and applications have their own set of install instructions. You should follow these in most cases.

### News

The 1.0 release of the Zope Toolkit contains a number of refactorings that are aimed to clean up dependencies between pieces of code. Many packages in `zope.app` have had their code moved to existing or newly created packages in the `zope` namespace. These new packages do generally not contain user interface code (typically what's in `.browser`), and have much clearer dependency relationships as a result.

Backwards compatibility imports have been left in place so that your existing code should still work. In some cases you will have to explicitly add dependencies to a `zope.app.` to your code, as due to the cleanup they might not come in automatically anymore due to indirect dependencies; if you see an import error this is probably the case.

We recommend you update your existing code to import from the new packages if possible. The transition support and `zope.app.*` support is limited: the legacy support will be officially retired from the ZTK after January 1, 2011.

### Migration issues

#### zope.app.keyreference -> zope.keyreference

This package was renamed to `zope.keyreference` and all its functionality was moved to the new one. The new package contains a little workaround for making old persistent keyreferences loadable without `zope.app.keyreference` installed, so the latter one is not needed at all anymore. Still review your code for any imports coming from `zope.app.keyreference` and modify it to use `zope.keyreference` instead.

### zope.app.intid -> zope.intid

The non-UI functionality of these packages was moved to `zope.intid` with backwards compatibility imports left in place. Review your imports from `zope.app.intid` to see whether they cannot come directly from `zope.intid` instead.

### zope.app.catalog -> zope.catalog

The non-UI functionality of these packages was moved to `zope.catalog`. Review your imports from `zope.app.catalog` to see whether they cannot come directly from `zope.catalog` instead.

### zope.app.container -> zope.container

The non-UI functionality of these packages was moved to `zope.container`. Review your imports from `zope.app.container` to see whether they cannot come directly from `zope.container` instead.

In addition, the exceptions used by `zope.container` were changed, so if your code catches them, you need to review it:

- The `DuplicationError` in `setitem` was changed to `KeyError`.
- The `UserError` in `NameChooser` was changed to `ValueError`.

### zope.app.component -> zope.security, zope.site

The implementation of the `<class>` ZCML directive moved from this package to `zope.security`. Packages that relied on `zope.app.component` to obtain this directive should declare a direct dependency on `zope.security`, and it may be possible to lose the dependency on `zope.app.component` altogether.

Non-UI site related functionality has been moved to the `zope.site` package. with backwards compatibility imports left in place. Review your imports from `zope.app.component` to see whether they cannot come directly from `zope.site` instead.

### zope.app.folder -> zope.site, zope.container

The implementation of the `zope.app.folder.Folder` class has moved to `zope.site.folder` instead, with backwards compatibility imports left in place. Review your imports from `zope.app.folder` to see whether they cannot come directly from `zope.site` instead. In addition, `Folder` is an `IContainer` implementation that also mixes in site management functionality. If such site management support is not necessary, in some cases your code does not need `Folder` but may be able to rely on a `Container` implementation from `zope.container` instead.

A base class with the implementation of the container-like behavior of `Folder` has moved to `zope.container` (and `zope.site` uses this for its implementation of `Folder`). This is not normally something you should need to retain backwards compatibility.

### zc.copy -> zope.copy, zope.copypastemove, zope.location

The pluggable object copying mechanism once developed in the `zc.copy` package was merged back into `zope.location`, `zope.copypastemove` and the new `zope.copy` package. The `zope.copy` package now provides a pluggable mechanism for copying objects from `zc.copy` and doesn't depend on anything but `zope.interface`. The `zope.copypastemove` uses the `copy` function from `zope.copy` in its `ObjectCopier`.

The zope.location now provides an ICopyHook adapter that implements conditional copy functionality based on object locations, that old zope.location.pickling.CopyPersistent used to provide. Note, that if you don't use ZCML configuration of zope.location, you may need to register zope.location.pickling.LocationCopyHook yourself.

The zope.location.pickling.locationCopy and zope.location.pickling.CopyPersistent are now deprecated in favor of zope.copy and were replaced by backward-compatibility imports. See zope.copy package documentation for information on how to use the new mechanism.

The new version of the zc.copy package now only contains backward-compatibility imports and is deprecated. zope.copy should be preferred for new developments.

### zope.app.security refactoring

The zope.app.security package was finally refactored into a few small parts with less dependencies and more clear purpose.

The implementation of the <module> ZCML directive moved from this package to zope.security. Packages that relied on zope.app.security to obtain this directive should declare a direct dependency on zope.security, and it may be possible to lose the dependency on zope.app.security altogether.

The protectclass module in this package has moved to zope.security, with backwards compatibility imports left in place. Review your imports from zope.app.security to see whether they cannot come directly from zope.security instead.

All interfaces (*IAuthentication*, *IUnauthenticatedPrincipal*, *ILoginPassword* and so on.) were moved into a new zope.authentication package, as well as several utility things, like *PrincipalSource* and *checkPrincipal* function. The new package has much less dependencies and defines an abstract contract for implementing authentication policies. While backward compatibility imports are left in place, it's strongly recommended to update your imports to the zope.authentication.

The *global principal registry* and its ZCML directives are moved into a new zope.principalregistry package with backward-compatibility imports left in place. If your application uses global principals, review your code and ZCML configuration to update it to the new place.

The *local permission* functionality was moved into a new zope.app.localpermission package. This functionality is a part of Through-The-Web development pattern that seems not to be used and supported much by Zope Toolkit and Application anymore, so it can be considered deprecated. However, it can serve as a great example of TTW-related component.

The *Permission vocabularies* and standard protections for Message objects and *__name__*, *__parent__* attributes as well as some common permissions, like *zope.View* and *zope.ManageContent* were merged into *zope.security*.

The adapters from zope.publisher's *IHTTPCredentials* and *IFTPCredentials* to the *ILoginPassword* were moved into zope.publisher, thus making zope.authentication a dependency for zope.publisher.

The original zope.app.security package now only contains several deprecated or application-specific permission definitions, python module protections, that are only likely to be needed with deprecated Through-The-Web development pattern, and ZMI-related browser views (login.html, zope.app.form view for PrincipalSource and so on), as well as backward-compatibility imports. So, if you're not using TTW and/or standard ZMI browser views, you probably should review update your imports to a new places and drop dependency on zope.app.security to reduce package dependencies count.

Other packages, that used zope.app.security, like zope.securitypolicy are either already adapted to the changes or will be adapted soon.

### zope.app.publisher refactoring

The `zope.app.publisher` package was also refactored into smaller parts with less dependencies and clearer purpose.

The browser resources mechanism (mostly used for serving static files and directories) was factored out to the new `zope.browserresource` package. It was also made more pluggable, so you can register specific resource classes for some file extensions, if you need special processing. One of the example is the new `zope.ptresource` package, where the PageTemplateResource was moved, another example is `z3c.zrtresource` package that was adapted to automatically use ZRT resource class for files with `.zrt` extensions.

Browser menu mechanism was moved into a new `zope.browsermenu` package with no further changes.

ZCML directives for easy creation of browser views (the `browser:page` directive and friends) was moved into a new small package, `zope.browserpage`. Also, the directives don't depend the menu mechanism now and will simply ignore "menu" and "title" arguments if `zope.browsermenu` package is not installed.

The `IModifiableBrowserLanguages` adapter was moved into `zope.publisher` along with several ZCML security declarations for `zope.publisher` classes that used to be in `zope.app.publisher`.

ZCML registrations for `IXMLRPCPublisher` adapter for containers was moved into the `zope.container`, because the actual adapters code were already in `zope.container` and registered there as `IBrowserPublisher` adapters. However, both adapters and their ZCML registrations will probably move elsewhere when we'll be refactoring `zope.container`.

Several parts are left in `zope.app.publisher` untouched:

- `Browser Skins` vocabulary.
- `date` field converter for `zope.publisher`'s form values conversion mechanism.
- `ManagementViewSelector` browser view (ZMI-related part).
- `xmlrpc:view` directive for publishing XML-RPC methods.

The latter, `xmlrpc:view` directive is generally useful, so it may be moved into a separate package in future, however there are no clear decision about how to move XML-RPC and FTP-related things currently.

### Password managers extracted from zope.app.authentication

The *IPasswordManager* interface and its implementations were extracted from `zope.app.authentication` into a new `zope.password` package to make them usable with other authentication systems, like `z3c.authenticator` or `zope.principalregistry` or any custom one.

It basically depends only on `zope.interface`, so it can be really useful even in non-Zope environments, like `Pylons`, for example.

The *Password Manager Names* vocabulary is also moved into `zope.password`, however, it's only useful with `zope.schema` and `zope.component`, so you need them installed to work with them. They're listed in the "vocabulary" extra requirement specification.

### ZODB 3.9 FileStorage native blob support

The FileStorage component of ZODB 3.9 used in Zope Toolkit 1.0 now supports blobs natively, so you don't need to use BlobStorage proxy for it anymore.

Thus, you can specify blob directory directly to FileStorage. If you use ZConfig, that means something like this:

```
<filestorage>
  path var/Data.fs
  blob-dir var/blobs
</filestorage>
```

instead of:

```
<blobstorage>
  blob-dir var/blobs
  <filestorage>
    path var/Data.fs
  </filestorage>
</blobstorage>
```

If you creating a storage from python, that means something like this:

```
storage = FileStorage('var/Data.fs', blob_dir='var/blobs')
```

instead of:

```
storage = BlobStorage('var/blobs', FileStorage('var/Data.fs'))
```

### Zope Toolkit 1.0.8

This document covers major changes in this release that can lead to backward-incompatibilities and explains what to look out for when updating.

#### Introduction

The Zope Toolkit 1.0 release is the first release of the Zope Toolkit. The Zope Toolkit really is just a collection of libraries managed together by the Zope developers. We typically treat each library independently, so you would like to look at the CHANGES.txt in each library for updates. Here we note larger changes, especially ones that affect multiple libraries.

#### Installation

The Zope Toolkit cannot be installed directly except as individual libraries (such as `zope.component`). To install it you typically would install a framework or application that makes use of these libraries. Examples of such projects are BlueBream, Grok and Zope 2.

If you want to use the Zope Toolkit KGS, you can use the buildout extends mechanism (replace 1.0 by the desired version):

```
[buildout]
extends = http://download.zope.org/zopetoolkit/index/1.0/ztk-versions.cfg
```

You can also copy the file locally or additionally extend the zopeapp-versions.cfg file from the same location.

Frameworks and applications have their own set of install instructions. You should follow these in most cases.

### Python versions

The ZTK 1.0 release series supports Python 2.4 up to Python 2.6. Neither Python 2.7 nor any Python 3.x series is supported.

### News

The 1.0 release of the Zope Toolkit contains a number of refactorings that are aimed to clean up dependencies between pieces of code. Many packages in `zope.app` have had their code moved to existing or newly created packages in the `zope` namespace. These new packages do generally not contain user interface code (typically what's in `.browser`), and have much clearer dependency relationships as a result.

Backwards compatibility imports have been left in place so that your existing code should still work. In some cases you will have to explicitly add dependencies to a `zope.app.` to your code, as due to the cleanup they might not come in automatically anymore due to indirect dependencies; if you see an import error this is probably the case.

We recommend you update your existing code to import from the new packages if possible. The transition support and `zope.app.*` support is limited: the legacy support will be officially retired from the ZTK in subsequent releases.

### Migration issues

#### zope.app.keyreference -> zope.keyreference

This package was renamed to `zope.keyreference` and all its functionality was moved to the new one. The new package contains a little workaround for making old persistent keyreferences loadable without `zope.app.keyreference` installed, so the latter one is not needed at all anymore. Still review your code for any imports coming from `zope.app.keyreference` and modify it to use `zope.keyreference` instead.

#### zope.app.intid -> zope.intid

The non-UI functionality of these packages was moved to `zope.intid` with backwards compatibility imports left in place. Review your imports from `zope.app.intid` to see whether they cannot come directly from `zope.intid` instead.

#### zope.app.catalog -> zope.catalog

The non-UI functionality of these packages was moved to `zope.catalog`. Review your imports from `zope.app.catalog` to see whether they cannot come directly from `zope.catalog` instead.

#### zope.app.container -> zope.container

The non-UI functionality of these packages was moved to `zope.container`. Review your imports from `zope.app.container` to see whether they cannot come directly from `zope.container` instead.

In addition, the exceptions used by `zope.container` were changed, so if your code catches them, you need to review it:

- The `DuplicationError` in `setitem` was changed to `KeyError`.
- The `UserError` in `NameChooser` was changed to `ValueError`.

### zope.app.component -> zope.security, zope.site

The implementation of the <class> ZCML directive moved from this package to zope.security. Packages that relied on zope.app.component to obtain this directive should declare a direct dependency on zope.security, and it may be possible to lose the dependency on zope.app.component altogether.

Non-UI site related functionality has been moved to the zope.site package. with backwards compatibility imports left in place. Review your imports from zope.app.component to see whether they cannot come directly from zope.site instead.

### zope.app.folder -> zope.site, zope.container

The implementation of the zope.app.folder.Folder class has moved to zope.site.folder instead, with backwards compatibility imports left in place. Review your imports from zope.app.folder to see whether they cannot come directly from zope.site instead. In addition, Folder is an IContainer implementation that also mixes in site management functionality. If such site management support is not necessary, in some cases your code does not need Folder but may be able to rely on a Container implementation from zope.container instead.

A base class with the implementation of the container-like behavior of Folder has moved to zope.container (and zope.site uses this for its implementation of Folder). This is not normally something you should need to retain backwards compatibility.

### zc.copy -> zope.copy, zope.copypastemove, zope.location

The pluggable object copying mechanism once developed in the zc.copy package was merged back into zope.location, zope.copypastemove and the new zope.copy package. The zope.copy package now provides a pluggable mechanism for copying objects from zc.copy and doesn't depend on anything but zope.interface. The zope.copypastemove uses the copy function from zope.copy in its ObjectCopier.

The zope.location now provides an ICopyHook adapter that implements conditional copy functionality based on object locations, that old zope.location.pickling.CopyPersistent used to provide. Note, that if you don't use ZCML configuration of zope.location, you may need to register zope.location.pickling.LocationCopyHook yourself.

The zope.location.pickling.locationCopy and zope.location.pickling.CopyPersistent are now deprecated in favor of zope.copy and were replaced by backward-compatibility imports. See zope.copy package documentation for information on how to use the new mechanism.

The new version of the zc.copy package now only contains backward-compatibility imports and is deprecated. zope.copy should be preferred for new developments.

### zope.app.security refactoring

The zope.app.security package was finally refactored into a few small parts with less dependencies and more clear purpose.

The implementation of the <module> ZCML directive moved from this package to zope.security. Packages that relied on zope.app.security to obtain this directive should declare a direct dependency on zope.security, and it may be possible to lose the dependency on zope.app.security altogether.

The protectclass module in this package has moved to zope.security, with backwards compatibility imports left in place. Review your imports from zope.app.security to see whether they cannot come directly from zope.security instead.

All interfaces (*IAuthentication*, *IUnauthenticatedPrincipal*, *ILoginPassword* and so on.) were moved into a new `zope.authentication` package, as well as several utility things, like *PrincipalSource* and *checkPrincipal* function. The new package has much less dependencies and defines an abstract contract for implementing authentication policies. While backward compatibility imports are left in place, it's strongly recommended to update your imports to the `zope.authentication`.

The *global principal registry* and its ZCML directives are moved into a new `zope.principalregistry` package with backward-compatibility imports left in place. If your application uses global principals, review your code and ZCML configuration to update it to the new place.

The *local permission* functionality was moved into a new `zope.app.localpermission` package. This functionality is a part of Through-The-Web development pattern that seems not to be used and supported much by Zope Toolkit and Application anymore, so it can be considered deprecated. However, it can serve as a great example of TTW-related component.

The *Permission vocabularies* and standard protections for Message objects and *__name__*, *__parent__* attributes as well as some common permissions, like *zope.View* and *zope.ManageContent* were merged into *zope.security*.

The adapters from `zope.publisher`'s *IHTTPCredentials* and *IFTPCredentials* to the *ILoginPassword* were moved into `zope.publisher`, thus making `zope.authentication` a dependency for `zope.publisher`.

The original `zope.app.security` package now only contains several deprecated or application-specific permission definitions, python module protections, that are only likely to be needed with deprecated Through-The-Web development pattern, and ZMI-related browser views (login.html, zope.app.form view for PrincipalSource and so on), as well as backward-compatibility imports. So, if you're not using TTW and/or standard ZMI browser views, you probably should review update your imports to a new places and drop dependency on `zope.app.security` to reduce package dependencies count.

Other packages, that used `zope.app.security`, like `zope.securitypolicy` are either already adapted to the changes or will be adapted soon.

### zope.app.publisher refactoring

The `zope.app.publisher` package was also refactored into smaller parts with less dependencies and clearer purpose.

The browser resources mechanism (mostly used for serving static files and directories) was factored out to the new `zope.browserresource` package. It was also made more pluggable, so you can register specific resource classes for some file extensions, if you need special processing. One of the example is the new `zope.ptresource` package, where the PageTemplateResource was moved, another example is `z3c.zrtresource` package that was adapted to automatically use ZRT resource class for files with `.zrt` extensions.

Browser menu mechanism was moved into a new `zope.browsermenu` package with no further changes.

ZCML directives for easy creation of browser views (the `browser:page` directive and friends) was moved into a new small package, `zope.browserpage`. Also, the directives don't depend the menu mechanism now and will simply ignore "menu" and "title" arguments if `zope.browsermenu` package is not installed.

The `IModifiableBrowserLanguages` adapter was moved into `zope.publisher` along with several ZCML security declarations for `zope.publisher` classes that used to be in `zope.app.publisher`.

ZCML registrations for `IXMLRPCPublisher` adapter for containers was moved into the `zope.container`, because the actual adapters code were already in `zope.container` and registered there as `IBrowserPublisher` adapters. However, both adapters and their ZCML registrations will probably move elsewhere when we'll be refactoring `zope.container`.

Several parts are left in `zope.app.publisher` untouched:

- `Browser Skins` vocabulary.

- `date` field converter for `zope.publisher`'s form values conversion mechanism.
- `ManagementViewSelector` browser view (ZMI-related part).
- `xmlrpc:view` directive for publishing XML-RPC methods.

The latter, `xmlrpc:view` directive is generally useful, so it may be moved into a separate package in future, however there are no clear decision about how to move XML-RPC and FTP-related things currently.

### Password managers extracted from zope.app.authentication

The *IPasswordManager* interface and its implementations were extracted from `zope.app.authentication` into a new `zope.password` package to make them usable with other authentication systems, like `z3c.authenticator` or `zope.principalregistry` or any custom one.

It basically depends only on `zope.interface`, so it can be really useful even in non-Zope environments, like `Pylons`, for example.

The *Password Manager Names* vocabulary is also moved into `zope.password`, however, it's only useful with `zope.schema` and `zope.component`, so you need them installed to work with them. They're listed in the "vocabulary" extra requirement specification.

### ZODB 3.9 FileStorage native blob support

The FileStorage component of ZODB 3.9 used in Zope Toolkit 1.0 now supports blobs natively, so you don't need to use BlobStorage proxy for it anymore.

Thus, you can specify blob directory directly to FileStorage. If you use ZConfig, that means something like this:

```
<filestorage>
  path var/Data.fs
  blob-dir var/blobs
</filestorage>
```

instead of:

```
<blobstorage>
  blob-dir var/blobs
  <filestorage>
    path var/Data.fs
  </filestorage>
</blobstorage>
```

If you creating a storage from python, that means something like this:

```
storage = FileStorage('var/Data.fs', blob_dir='var/blobs')
```

instead of:

```
storage = BlobStorage('var/blobs', FileStorage('var/Data.fs'))
```

### zope.dublincore permission renaming

The `zope.app.dublincore.*` permissions have been renamed to `zope.dublincore.*`. Applications using these permissions have to fix up grants based on the old permissions.

### Zope Toolkit 1.0.7

This document covers major changes in this release that can lead to backward-incompatibilities and explains what to look out for when updating.

#### Introduction

The Zope Toolkit 1.0 release is the first release of the Zope Toolkit. The Zope Toolkit really is just a collection of libraries managed together by the Zope developers. We typically treat each library independently, so you would like to look at the CHANGES.txt in each library for updates. Here we note larger changes, especially ones that affect multiple libraries.

#### Installation

The Zope Toolkit cannot be installed directly except as individual libraries (such as `zope.component`). To install it you typically would install a framework or application that makes use of these libraries. Examples of such projects are BlueBream, Grok and Zope 2.

If you want to use the Zope Toolkit KGS, you can use the buildout extends mechanism (replace 1.0 by the desired version):

```
[buildout]
extends = http://download.zope.org/zopetoolkit/index/1.0/ztk-versions.cfg
```

You can also copy the file locally or additionally extend the zopeapp-versions.cfg file from the same location.

Frameworks and applications have their own set of install instructions. You should follow these in most cases.

#### Python versions

The ZTK 1.0 release series supports Python 2.4 up to Python 2.6. Neither Python 2.7 nor any Python 3.x series is supported.

#### News

The 1.0 release of the Zope Toolkit contains a number of refactorings that are aimed to clean up dependencies between pieces of code. Many packages in `zope.app` have had their code moved to existing or newly created packages in the `zope` namespace. These new packages do generally not contain user interface code (typically what's in `.browser`), and have much clearer dependency relationships as a result.

Backwards compatibility imports have been left in place so that your existing code should still work. In some cases you will have to explicitly add dependencies to a `zope.app.` to your code, as due to the cleanup they might not come in automatically anymore due to indirect dependencies; if you see an import error this is probably the case.

We recommend you update your existing code to import from the new packages if possible. The transition support and `zope.app.*` support is limited: the legacy support will be officially retired from the ZTK in subsequent releases.

#### Migration issues

### zope.app.keyreference -> zope.keyreference

This package was renamed to `zope.keyreference` and all its functionality was moved to the new one. The new package contains a little workaround for making old persistent keyreferences loadable without `zope.app.keyreference` installed, so the latter one is not needed at all anymore. Still review your code for any imports coming from `zope.app.keyreference` and modify it to use `zope.keyreference` instead.

### zope.app.intid -> zope.intid

The non-UI functionality of these packages was moved to `zope.intid` with backwards compatibility imports left in place. Review your imports from `zope.app.intid` to see whether they cannot come directly from `zope.intid` instead.

### zope.app.catalog -> zope.catalog

The non-UI functionality of these packages was moved to `zope.catalog`. Review your imports from `zope.app.catalog` to see whether they cannot come directly from `zope.catalog` instead.

### zope.app.container -> zope.container

The non-UI functionality of these packages was moved to `zope.container`. Review your imports from `zope.app.container` to see whether they cannot come directly from `zope.container` instead.

In addition, the exceptions used by `zope.container` were changed, so if your code catches them, you need to review it:

- The `DuplicationError` in `setitem` was changed to `KeyError`.
- The `UserError` in `NameChooser` was changed to `ValueError`.

### zope.app.component -> zope.security, zope.site

The implementation of the `<class>` ZCML directive moved from this package to `zope.security`. Packages that relied on `zope.app.component` to obtain this directive should declare a direct dependency on `zope.security`, and it may be possible to lose the dependency on `zope.app.component` altogether.

Non-UI site related functionality has been moved to the `zope.site` package. with backwards compatibility imports left in place. Review your imports from `zope.app.component` to see whether they cannot come directly from `zope.site` instead.

### zope.app.folder -> zope.site, zope.container

The implementation of the `zope.app.folder.Folder` class has moved to `zope.site.folder` instead, with backwards compatibility imports left in place. Review your imports from `zope.app.folder` to see whether they cannot come directly from `zope.site` instead. In addition, `Folder` is an `IContainer` implementation that also mixes in site management functionality. If such site management support is not necessary, in some cases your code does not need `Folder` but may be able to rely on a `Container` implementation from `zope.container` instead.

A base class with the implementation of the container-like behavior of `Folder` has moved to `zope.container` (and `zope.site` uses this for its implementation of `Folder`). This is not normally something you should need to retain backwards compatibility.

### zc.copy -> zope.copy, zope.copypastemove, zope.location

The pluggable object copying mechanism once developed in the `zc.copy` package was merged back into `zope.location`, `zope.copypastemove` and the new `zope.copy` package. The `zope.copy` package now provides a pluggable mechanism for copying objects from `zc.copy` and doesn't depend on anything but `zope.interface`. The `zope.copypastemove` uses the `copy` function from `zope.copy` in its `ObjectCopier`.

The `zope.location` now provides an `ICopyHook` adapter that implements conditional copy functionality based on object locations, that old `zope.location.pickling.CopyPersistent` used to provide. Note, that if you don't use ZCML configuration of `zope.location`, you may need to register `zope.location.pickling.LocationCopyHook` yourself.

The `zope.location.pickling.locationCopy` and `zope.location.pickling.CopyPersistent` are now deprecated in favor of `zope.copy` and were replaced by backward-compatibility imports. See `zope.copy` package documentation for information on how to use the new mechanism.

The new version of the `zc.copy` package now only contains backward-compatibility imports and is deprecated. `zope.copy` should be preferred for new developments.

### zope.app.security refactoring

The `zope.app.security` package was finally refactored into a few small parts with less dependencies and more clear purpose.

The implementation of the `<module>` ZCML directive moved from this package to `zope.security`. Packages that relied on `zope.app.security` to obtain this directive should declare a direct dependency on `zope.security`, and it may be possible to lose the dependency on `zope.app.security` altogether.

The `protectclass` module in this package has moved to `zope.security`, with backwards compatibility imports left in place. Review your imports from `zope.app.security` to see whether they cannot come directly from `zope.security` instead.

All interfaces (*IAuthentication*, *IUnauthenticatedPrincipal*, *ILoginPassword* and so on.) were moved into a new `zope.authentication` package, as well as several utility things, like *PrincipalSource* and *checkPrincipal* function. The new package has much less dependencies and defines an abstract contract for implementing authentication policies. While backward compatibility imports are left in place, it's strongly recommended to update your imports to the `zope.authentication`.

The *global principal registry* and its ZCML directives are moved into a new `zope.principalregistry` package with backward-compatibility imports left in place. If your application uses global principals, review your code and ZCML configuration to update it to the new place.

The *local permission* functionality was moved into a new `zope.app.localpermission` package. This functionality is a part of Through-The-Web development pattern that seems not to be used and supported much by Zope Toolkit and Application anymore, so it can be considered deprecated. However, it can serve as a great example of TTW-related component.

The *Permission vocabularies* and standard protections for Message objects and *__name__*, *__parent__* attributes as well as some common permissions, like *zope.View* and *zope.ManageContent* were merged into *zope.security*.

The adapters from `zope.publisher`'s *IHTTPCredentials* and *IFTPCredentials* to the *ILoginPassword* were moved into `zope.publisher`, thus making `zope.authentication` a dependency for `zope.publisher`.

The original `zope.app.security` package now only contains several deprecated or application-specific permission definitions, python module protections, that are only likely to be needed with deprecated Through-The-Web development pattern, and ZMI-related browser views (login.html, zope.app.form view for PrincipalSource and so on), as well as backward-compatibility imports. So, if you're not using TTW and/or standard ZMI browser views, you

probably should review update your imports to a new places and drop dependency on `zope.app.security` to reduce package dependencies count.

Other packages, that used `zope.app.security`, like `zope.securitypolicy` are either already adapted to the changes or will be adapted soon.

### zope.app.publisher refactoring

The `zope.app.publisher` package was also refactored into smaller parts with less dependencies and clearer purpose.

The browser resources mechanism (mostly used for serving static files and directories) was factored out to the new `zope.browserresource` package. It was also made more pluggable, so you can register specific resource classes for some file extensions, if you need special processing. One of the example is the new `zope.ptresource` package, where the PageTemplateResource was moved, another example is `z3c.zrtresource` package that was adapted to automatically use ZRT resource class for files with `.zrt` extensions.

Browser menu mechanism was moved into a new `zope.browsermenu` package with no further changes.

ZCML directives for easy creation of browser views (the `browser:page` directive and friends) was moved into a new small package, `zope.browserpage`. Also, the directives don't depend the menu mechanism now and will simply ignore "menu" and "title" arguments if `zope.browsermenu` package is not installed.

The `IModifiableBrowserLanguages` adapter was moved into `zope.publisher` along with several ZCML security declarations for `zope.publisher` classes that used to be in `zope.app.publisher`.

ZCML registrations for `IXMLRPCPublisher` adapter for containers was moved into the `zope.container`, because the actual adapters code were already in `zope.container` and registered there as `IBrowserPublisher` adapters. However, both adapters and their ZCML registrations will probably move elsewhere when we'll be refactoring `zope.container`.

Several parts are left in `zope.app.publisher` untouched:

- `Browser Skins` vocabulary.
- `date` field converter for `zope.publisher`'s form values conversion mechanism.
- `ManagementViewSelector` browser view (ZMI-related part).
- `xmlrpc:view` directive for publishing XML-RPC methods.

The latter, `xmlrpc:view` directive is generally useful, so it may be moved into a separate package in future, however there are no clear decision about how to move XML-RPC and FTP-related things currently.

### Password managers extracted from zope.app.authentication

The *IPasswordManager* interface and its implementations were extracted from `zope.app.authentication` into a new `zope.password` package to make them usable with other authentication systems, like `z3c.authenticator` or `zope.principalregistry` or any custom one.

It basically depends only on `zope.interface`, so it can be really useful even in non-Zope environments, like `Pylons`, for example.

The *Password Manager Names* vocabulary is also moved into `zope.password`, however, it's only useful with `zope.schema` and `zope.component`, so you need them installed to work with them. They're listed in the "vocabulary" extra requirement specification.

### ZODB 3.9 FileStorage native blob support

The FileStorage component of ZODB 3.9 used in Zope Toolkit 1.0 now supports blobs natively, so you don't need to use BlobStorage proxy for it anymore.

Thus, you can specify blob directory directly to FileStorage. If you use ZConfig, that means something like this:

```
<filestorage>
  path var/Data.fs
  blob-dir var/blobs
</filestorage>
```

instead of:

```
<blobstorage>
  blob-dir var/blobs
  <filestorage>
    path var/Data.fs
  </filestorage>
</blobstorage>
```

If you creating a storage from python, that means something like this:

```
storage = FileStorage('var/Data.fs', blob_dir='var/blobs')
```

instead of:

```
storage = BlobStorage('var/blobs', FileStorage('var/Data.fs'))
```

### zope.dublincore permission renaming

The `zope.app.dublincore.*` permissions have been renamed to `zope.dublincore.*`. Applications using these permissions have to fix up grants based on the old permissions.

### Zope Toolkit 1.0.6

This document covers major changes in this release that can lead to backward-incompatibilities and explains what to look out for when updating.

### Introduction

The Zope Toolkit 1.0 release is the first release of the Zope Toolkit. The Zope Toolkit really is just a collection of libraries managed together by the Zope developers. We typically treat each library independently, so you would like to look at the CHANGES.txt in each library for updates. Here we note larger changes, especially ones that affect multiple libraries.

### Installation

The Zope Toolkit cannot be installed directly except as individual libraries (such as `zope.component`). To install it you typically would install a framework or application that makes use of these libraries. Examples of such projects are BlueBream, Grok and Zope 2.

If you want to use the Zope Toolkit KGS, you can use the buildout extends mechanism (replace 1.0 by the desired version):

```
[buildout]
extends = http://download.zope.org/zopetoolkit/index/1.0/ztk-versions.cfg
```

You can also copy the file locally or additionally extend the zopeapp-versions.cfg file from the same location.

Frameworks and applications have their own set of install instructions. You should follow these in most cases.

### Python versions

The ZTK 1.0 release series supports Python 2.4 up to Python 2.6. Neither Python 2.7 nor any Python 3.x series is supported.

### News

The 1.0 release of the Zope Toolkit contains a number of refactorings that are aimed to clean up dependencies between pieces of code. Many packages in `zope.app` have had their code moved to existing or newly created packages in the `zope` namespace. These new packages do generally not contain user interface code (typically what's in `.browser`), and have much clearer dependency relationships as a result.

Backwards compatibility imports have been left in place so that your existing code should still work. In some cases you will have to explicitly add dependencies to a `zope.app.` to your code, as due to the cleanup they might not come in automatically anymore due to indirect dependencies; if you see an import error this is probably the case.

We recommend you update your existing code to import from the new packages if possible. The transition support and `zope.app.*` support is limited: the legacy support will be officially retired from the ZTK in subsequent releases.

### Migration issues

#### zope.app.keyreference -> zope.keyreference

This package was renamed to `zope.keyreference` and all its functionality was moved to the new one. The new package contains a little workaround for making old persistent keyreferences loadable without `zope.app.keyreference` installed, so the latter one is not needed at all anymore. Still review your code for any imports coming from `zope.app.keyreference` and modify it to use `zope.keyreference` instead.

#### zope.app.intid -> zope.intid

The non-UI functionality of these packages was moved to `zope.intid` with backwards compatibility imports left in place. Review your imports from `zope.app.intid` to see whether they cannot come directly from `zope.intid` instead.

#### zope.app.catalog -> zope.catalog

The non-UI functionality of these packages was moved to `zope.catalog`. Review your imports from `zope.app.catalog` to see whether they cannot come directly from `zope.catalog` instead.

### zope.app.container -> zope.container

The non-UI functionality of these packages was moved to `zope.container`. Review your imports from `zope.app.container` to see whether they cannot come directly from `zope.container` instead.

In addition, the exceptions used by `zope.container` were changed, so if your code catches them, you need to review it:

- The `DuplicationError` in `setitem` was changed to `KeyError`.
- The `UserError` in `NameChooser` was changed to `ValueError`.

### zope.app.component -> zope.security, zope.site

The implementation of the `<class>` ZCML directive moved from this package to `zope.security`. Packages that relied on `zope.app.component` to obtain this directive should declare a direct dependency on `zope.security`, and it may be possible to lose the dependency on `zope.app.component` altogether.

Non-UI site related functionality has been moved to the `zope.site` package. with backwards compatibility imports left in place. Review your imports from `zope.app.component` to see whether they cannot come directly from `zope.site` instead.

### zope.app.folder -> zope.site, zope.container

The implementation of the `zope.app.folder.Folder` class has moved to `zope.site.folder` instead, with backwards compatibility imports left in place. Review your imports from `zope.app.folder` to see whether they cannot come directly from `zope.site` instead. In addition, `Folder` is an `IContainer` implementation that also mixes in site management functionality. If such site management support is not necessary, in some cases your code does not need `Folder` but may be able to rely on a `Container` implementation from `zope.container` instead.

A base class with the implementation of the container-like behavior of `Folder` has moved to `zope.container` (and `zope.site` uses this for its implementation of `Folder`). This is not normally something you should need to retain backwards compatibility.

### zc.copy -> zope.copy, zope.copypastemove, zope.location

The pluggable object copying mechanism once developed in the `zc.copy` package was merged back into `zope.location`, `zope.copypastemove` and the new `zope.copy` package. The `zope.copy` package now provides a pluggable mechanism for copying objects from `zc.copy` and doesn't depend on anything but `zope.interface`. The `zope.copypastemove` uses the `copy` function from `zope.copy` in its `ObjectCopier`.

The `zope.location` now provides an `ICopyHook` adapter that implements conditional copy functionality based on object locations, that old `zope.location.pickling.CopyPersistent` used to provide. Note, that if you don't use ZCML configuration of `zope.location`, you may need to register `zope.location.pickling.LocationCopyHook` yourself.

The `zope.location.pickling.locationCopy` and `zope.location.pickling.CopyPersistent` are now deprecated in favor of `zope.copy` and were replaced by backward-compatibility imports. See `zope.copy` package documentation for information on how to use the new mechanism.

The new version of the `zc.copy` package now only contains backward-compatibility imports and is deprecated. `zope.copy` should be preferred for new developments.

### zope.app.security refactoring

The `zope.app.security` package was finally refactored into a few small parts with less dependencies and more clear purpose.

The implementation of the `<module>` ZCML directive moved from this package to `zope.security`. Packages that relied on `zope.app.security` to obtain this directive should declare a direct dependency on `zope.security`, and it may be possible to lose the dependency on `zope.app.security` altogether.

The `protectclass` module in this package has moved to `zope.security`, with backwards compatibility imports left in place. Review your imports from `zope.app.security` to see whether they cannot come directly from `zope.security` instead.

All interfaces (*IAuthentication*, *IUnauthenticatedPrincipal*, *ILoginPassword* and so on.) were moved into a new `zope.authentication` package, as well as several utility things, like *PrincipalSource* and *checkPrincipal* function. The new package has much less dependencies and defines an abstract contract for implementing authentication policies. While backward compatibility imports are left in place, it's strongly recommended to update your imports to the `zope.authentication`.

The *global principal registry* and its ZCML directives are moved into a new `zope.principalregistry` package with backward-compatibility imports left in place. If your application uses global principals, review your code and ZCML configuration to update it to the new place.

The *local permission* functionality was moved into a new `zope.app.localpermission` package. This functionality is a part of Through-The-Web development pattern that seems not to be used and supported much by Zope Toolkit and Application anymore, so it can be considered deprecated. However, it can serve as a great example of TTW-related component.

The *Permission vocabularies* and standard protections for Message objects and *__name__*, *__parent__* attributes as well as some common permissions, like *zope.View* and *zope.ManageContent* were merged into *zope.security*.

The adapters from `zope.publisher`'s *IHTTPCredentials* and *IFTPCredentials* to the *ILoginPassword* were moved into `zope.publisher`, thus making `zope.authentication` a dependency for `zope.publisher`.

The original `zope.app.security` package now only contains several deprecated or application-specific permission definitions, python module protections, that are only likely to be needed with deprecated Through-The-Web development pattern, and ZMI-related browser views (login.html, zope.app.form view for PrincipalSource and so on), as well as backward-compatibility imports. So, if you're not using TTW and/or standard ZMI browser views, you probably should review update your imports to a new places and drop dependency on `zope.app.security` to reduce package dependencies count.

Other packages, that used `zope.app.security`, like `zope.securitypolicy` are either already adapted to the changes or will be adapted soon.

### zope.app.publisher refactoring

The `zope.app.publisher` package was also refactored into smaller parts with less dependencies and clearer purpose.

The browser resources mechanism (mostly used for serving static files and directories) was factored out to the new `zope.browserresource` package. It was also made more pluggable, so you can register specific resource classes for some file extensions, if you need special processing. One of the example is the new `zope.ptresource` package, where the PageTemplateResource was moved, another example is `z3c.zrtresource` package that was adapted to automatically use ZRT resource class for files with `.zrt` extensions.

Browser menu mechanism was moved into a new `zope.browsermenu` package with no further changes.

ZCML directives for easy creation of browser views (the `browser:page` directive and friends) was moved into a new small package, `zope.browserpage`. Also, the directives don't depend the menu mechanism now and will simply ignore "menu" and "title" arguments if `zope.browsermenu` package is not installed.

The `IModifiableBrowserLanguages` adapter was moved into `zope.publisher` along with several ZCML security declarations for `zope.publisher` classes that used to be in `zope.app.publisher`.

ZCML registrations for `IXMLRPCPublisher` adapter for containers was moved into the `zope.container`, because the actual adapters code were already in `zope.container` and registered there as `IBrowserPublisher` adapters. However, both adapters and their ZCML registrations will probably move elsewhere when we'll be refactoring `zope.container`.

Several parts are left in `zope.app.publisher` untouched:

- `Browser Skins` vocabulary.
- `date` field converter for `zope.publisher`'s form values conversion mechanism.
- `ManagementViewSelector` browser view (ZMI-related part).
- `xmlrpc:view` directive for publishing XML-RPC methods.

The latter, `xmlrpc:view` directive is generally useful, so it may be moved into a separate package in future, however there are no clear decision about how to move XML-RPC and FTP-related things currently.

### Password managers extracted from zope.app.authentication

The *IPasswordManager* interface and its implementations were extracted from `zope.app.authentication` into a new `zope.password` package to make them usable with other authentication systems, like `z3c.authenticator` or `zope.principalregistry` or any custom one.

It basically depends only on `zope.interface`, so it can be really useful even in non-Zope environments, like `Pylons`, for example.

The *Password Manager Names* vocabulary is also moved into `zope.password`, however, it's only useful with `zope.schema` and `zope.component`, so you need them installed to work with them. They're listed in the "vocabulary" extra requirement specification.

### ZODB 3.9 FileStorage native blob support

The FileStorage component of ZODB 3.9 used in Zope Toolkit 1.0 now supports blobs natively, so you don't need to use BlobStorage proxy for it anymore.

Thus, you can specify blob directory directly to FileStorage. If you use ZConfig, that means something like this:

```
<filestorage>
  path var/Data.fs
  blob-dir var/blobs
</filestorage>
```

instead of:

```
<blobstorage>
  blob-dir var/blobs
  <filestorage>
    path var/Data.fs
  </filestorage>
</blobstorage>
```

If you creating a storage from python, that means something like this:

```
storage = FileStorage('var/Data.fs', blob_dir='var/blobs')
```

instead of:

```
storage = BlobStorage('var/blobs', FileStorage('var/Data.fs'))
```

### zope.dublincore permission renaming

The `zope.app.dublincore.*` permissions have been renamed to `zope.dublincore.*`. Applications using these permissions have to fix up grants based on the old permissions.

## Zope Toolkit 1.0.5

This document covers major changes in this release that can lead to backward-incompatibilities and explains what to look out for when updating.

### Introduction

The Zope Toolkit 1.0 release is the first release of the Zope Toolkit. The Zope Toolkit really is just a collection of libraries managed together by the Zope developers. We typically treat each library independently, so you would like to look at the CHANGES.txt in each library for updates. Here we note larger changes, especially ones that affect multiple libraries.

### Installation

The Zope Toolkit cannot be installed directly except as individual libraries (such as `zope.component`). To install it you typically would install a framework or application that makes use of these libraries. Examples of such projects are BlueBream, Grok and Zope 2.

If you want to use the Zope Toolkit KGS, you can use the buildout extends mechanism (replace 1.0 by the desired version):

```
[buildout]
extends = http://download.zope.org/zopetoolkit/index/1.0/ztk-versions.cfg
```

You can also copy the file locally or additionally extend the zopeapp-versions.cfg file from the same location.

Frameworks and applications have their own set of install instructions. You should follow these in most cases.

### Python versions

The ZTK 1.0 release series supports Python 2.4 up to Python 2.6. Neither Python 2.7 nor any Python 3.x series is supported.

### News

The 1.0 release of the Zope Toolkit contains a number of refactorings that are aimed to clean up dependencies between pieces of code. Many packages in `zope.app` have had their code moved to existing or newly created packages in the `zope` namespace. These new packages do generally not contain user interface code (typically what's in `.browser`), and have much clearer dependency relationships as a result.

Backwards compatibility imports have been left in place so that your existing code should still work. In some cases you will have to explicitly add dependencies to a `zope.app.` to your code, as due to the cleanup they might not come in automatically anymore due to indirect dependencies; if you see an import error this is probably the case.

We recommend you update your existing code to import from the new packages if possible. The transition support and `zope.app.*` support is limited: the legacy support will be officially retired from the ZTK in subsequent releases.

### Migration issues

#### zope.app.keyreference -> zope.keyreference

This package was renamed to `zope.keyreference` and all its functionality was moved to the new one. The new package contains a little workaround for making old persistent keyreferences loadable without `zope.app.keyreference` installed, so the latter one is not needed at all anymore. Still review your code for any imports coming from `zope.app.keyreference` and modify it to use `zope.keyreference` instead.

#### zope.app.intid -> zope.intid

The non-UI functionality of these packages was moved to `zope.intid` with backwards compatibility imports left in place. Review your imports from `zope.app.intid` to see whether they cannot come directly from `zope.intid` instead.

#### zope.app.catalog -> zope.catalog

The non-UI functionality of these packages was moved to `zope.catalog`. Review your imports from `zope.app.catalog` to see whether they cannot come directly from `zope.catalog` instead.

#### zope.app.container -> zope.container

The non-UI functionality of these packages was moved to `zope.container`. Review your imports from `zope.app.container` to see whether they cannot come directly from `zope.container` instead.

In addition, the exceptions used by `zope.container` were changed, so if your code catches them, you need to review it:

- The `DuplicationError` in `setitem` was changed to `KeyError`.
- The `UserError` in `NameChooser` was changed to `ValueError`.

#### zope.app.component -> zope.security, zope.site

The implementation of the `<class>` ZCML directive moved from this package to `zope.security`. Packages that relied on `zope.app.component` to obtain this directive should declare a direct dependency on `zope.security`, and it may be possible to lose the dependency on `zope.app.component` altogether.

Non-UI site related functionality has been moved to the `zope.site` package. with backwards compatibility imports left in place. Review your imports from `zope.app.component` to see whether they cannot come directly from `zope.site` instead.

### zope.app.folder -> zope.site, zope.container

The implementation of the `zope.app.folder.Folder` class has moved to `zope.site.folder` instead, with backwards compatibility imports left in place. Review your imports from `zope.app.folder` to see whether they cannot come directly from `zope.site` instead. In addition, `Folder` is an `IContainer` implementation that also mixes in site management functionality. If such site management support is not necessary, in some cases your code does not need `Folder` but may be able to rely on a `Container` implementation from `zope.container` instead.

A base class with the implementation of the container-like behavior of `Folder` has moved to `zope.container` (and `zope.site` uses this for its implementation of `Folder`). This is not normally something you should need to retain backwards compatibility.

### zc.copy -> zope.copy, zope.copypastemove, zope.location

The pluggable object copying mechanism once developed in the `zc.copy` package was merged back into `zope.location`, `zope.copypastemove` and the new `zope.copy` package. The `zope.copy` package now provides a pluggable mechanism for copying objects from `zc.copy` and doesn't depend on anything but `zope.interface`. The `zope.copypastemove` uses the `copy` function from `zope.copy` in its `ObjectCopier`.

The `zope.location` now provides an `ICopyHook` adapter that implements conditional copy functionality based on object locations, that old `zope.location.pickling.CopyPersistent` used to provide. Note, that if you don't use ZCML configuration of `zope.location`, you may need to register `zope.location.pickling.LocationCopyHook` yourself.

The `zope.location.pickling.locationCopy` and `zope.location.pickling.CopyPersistent` are now deprecated in favor of `zope.copy` and were replaced by backward-compatibility imports. See `zope.copy` package documentation for information on how to use the new mechanism.

The new version of the `zc.copy` package now only contains backward-compatibility imports and is deprecated. `zope.copy` should be preferred for new developments.

### zope.app.security refactoring

The `zope.app.security` package was finally refactored into a few small parts with less dependencies and more clear purpose.

The implementation of the `<module>` ZCML directive moved from this package to `zope.security`. Packages that relied on `zope.app.security` to obtain this directive should declare a direct dependency on `zope.security`, and it may be possible to lose the dependency on `zope.app.security` altogether.

The `protectclass` module in this package has moved to `zope.security`, with backwards compatibility imports left in place. Review your imports from `zope.app.security` to see whether they cannot come directly from `zope.security` instead.

All interfaces (*IAuthentication*, *IUnauthenticatedPrincipal*, *ILoginPassword* and so on.) were moved into a new `zope.authentication` package, as well as several utility things, like *PrincipalSource* and *checkPrincipal* function. The new package has much less dependencies and defines an abstract contract for implementing authentication policies. While backward compatibility imports are left in place, it's strongly recommended to update your imports to the `zope.authentication`.

The *global principal registry* and its ZCML directives are moved into a new `zope.principalregistry` package with backward-compatibility imports left in place. If your application uses global principals, review your code and ZCML configuration to update it to the new place.

The *local permission* functionality was moved into a new `zope.app.localpermission` package. This functionality is a part of Through-The-Web development pattern that seems not to be used and supported much by Zope Toolkit and Application anymore, so it can be considered deprecated. However, it can serve as a great example of TTW-related component.

The *Permission vocabularies* and standard protections for Message objects and *__name__*, *__parent__* attributes as well as some common permissions, like *zope.View* and *zope.ManageContent* were merged into *zope.security*.

The adapters from `zope.publisher`'s *IHTTPCredentials* and *IFTPCredentials* to the *ILoginPassword* were moved into `zope.publisher`, thus making `zope.authentication` a dependency for `zope.publisher`.

The original `zope.app.security` package now only contains several deprecated or application-specific permission definitions, python module protections, that are only likely to be needed with deprecated Through-The-Web development pattern, and ZMI-related browser views (login.html, zope.app.form view for PrincipalSource and so on), as well as backward-compatibility imports. So, if you're not using TTW and/or standard ZMI browser views, you probably should review update your imports to a new places and drop dependency on `zope.app.security` to reduce package dependencies count.

Other packages, that used `zope.app.security`, like `zope.securitypolicy` are either already adapted to the changes or will be adapted soon.

### zope.app.publisher refactoring

The `zope.app.publisher` package was also refactored into smaller parts with less dependencies and clearer purpose.

The browser resources mechanism (mostly used for serving static files and directories) was factored out to the new `zope.browserresource` package. It was also made more pluggable, so you can register specific resource classes for some file extensions, if you need special processing. One of the example is the new `zope.ptresource` package, where the PageTemplateResource was moved, another example is `z3c.zrtresource` package that was adapted to automatically use ZRT resource class for files with `.zrt` extensions.

Browser menu mechanism was moved into a new `zope.browsermenu` package with no further changes.

ZCML directives for easy creation of browser views (the `browser:page` directive and friends) was moved into a new small package, `zope.browserpage`. Also, the directives don't depend the menu mechanism now and will simply ignore "menu" and "title" arguments if `zope.browsermenu` package is not installed.

The `IModifiableBrowserLanguages` adapter was moved into `zope.publisher` along with several ZCML security declarations for `zope.publisher` classes that used to be in `zope.app.publisher`.

ZCML registrations for `IXMLRPCPublisher` adapter for containers was moved into the `zope.container`, because the actual adapters code were already in `zope.container` and registered there as `IBrowserPublisher` adapters. However, both adapters and their ZCML registrations will probably move elsewhere when we'll be refactoring `zope.container`.

Several parts are left in `zope.app.publisher` untouched:

- `Browser Skins` vocabulary.
- `date` field converter for `zope.publisher`'s form values conversion mechanism.
- `ManagementViewSelector` browser view (ZMI-related part).
- `xmlrpc:view` directive for publishing XML-RPC methods.

The latter, `xmlrpc:view` directive is generally useful, so it may be moved into a separate package in future, however there are no clear decision about how to move XML-RPC and FTP-related things currently.

### Password managers extracted from zope.app.authentication

The *IPasswordManager* interface and its implementations were extracted from `zope.app.authentication` into a new `zope.password` package to make them usable with other authentication systems, like `z3c.authenticator` or `zope.principalregistry` or any custom one.

It basically depends only on `zope.interface`, so it can be really useful even in non-Zope environments, like `Pylons`, for example.

The *Password Manager Names* vocabulary is also moved into `zope.password`, however, it's only useful with `zope.schema` and `zope.component`, so you need them installed to work with them. They're listed in the "vocabulary" extra requirement specification.

### ZODB 3.9 FileStorage native blob support

The FileStorage component of ZODB 3.9 used in Zope Toolkit 1.0 now supports blobs natively, so you don't need to use BlobStorage proxy for it anymore.

Thus, you can specify blob directory directly to FileStorage. If you use ZConfig, that means something like this:

```
<filestorage>
  path var/Data.fs
  blob-dir var/blobs
</filestorage>
```

instead of:

```
<blobstorage>
  blob-dir var/blobs
  <filestorage>
    path var/Data.fs
  </filestorage>
</blobstorage>
```

If you creating a storage from python, that means something like this:

```
storage = FileStorage('var/Data.fs', blob_dir='var/blobs')
```

instead of:

```
storage = BlobStorage('var/blobs', FileStorage('var/Data.fs'))
```

### zope.dublincore permission renaming

The `zope.app.dublincore.*` permissions have been renamed to `zope.dublincore.*`. Applications using these permissions have to fix up grants based on the old permissions.

### Zope Toolkit 1.0.4

This document covers major changes in this release that can lead to backward-incompatibilities and explains what to look out for when updating.

### Introduction

The Zope Toolkit 1.0 release is the first release of the Zope Toolkit. The Zope Toolkit really is just a collection of libraries managed together by the Zope developers. We typically treat each library independently, so you would like to look at the CHANGES.txt in each library for updates. Here we note larger changes, especially ones that affect multiple libraries.

### Installation

The Zope Toolkit cannot be installed directly except as individual libraries (such as `zope.component`). To install it you typically would install a framework or application that makes use of these libraries. Examples of such projects are BlueBream, Grok and Zope 2.

If you want to use the Zope Toolkit KGS, you can use the buildout extends mechanism (replace 1.0 by the desired version):

```
[buildout]
extends = http://download.zope.org/zopetoolkit/index/1.0/ztk-versions.cfg
```

You can also copy the file locally or additionally extend the zopeapp-versions.cfg file from the same location.

Frameworks and applications have their own set of install instructions. You should follow these in most cases.

### Python versions

The ZTK 1.0 release series supports Python 2.4 up to Python 2.6. Neither Python 2.7 nor any Python 3.x series is supported.

### News

The 1.0 release of the Zope Toolkit contains a number of refactorings that are aimed to clean up dependencies between pieces of code. Many packages in `zope.app` have had their code moved to existing or newly created packages in the `zope` namespace. These new packages do generally not contain user interface code (typically what's in `.browser`), and have much clearer dependency relationships as a result.

Backwards compatibility imports have been left in place so that your existing code should still work. In some cases you will have to explicitly add dependencies to a `zope.app.` to your code, as due to the cleanup they might not come in automatically anymore due to indirect dependencies; if you see an import error this is probably the case.

We recommend you update your existing code to import from the new packages if possible. The transition support and `zope.app.*` support is limited: the legacy support will be officially retired from the ZTK in subsequent releases.

### Migration issues

#### zope.app.keyreference -> zope.keyreference

This package was renamed to `zope.keyreference` and all its functionality was moved to the new one. The new package contains a little workaround for making old persistent keyreferences loadable without `zope.app.keyreference` installed, so the latter one is not needed at all anymore. Still review your code for any imports coming from `zope.app.keyreference` and modify it to use `zope.keyreference` instead.

### zope.app.intid -> zope.intid

The non-UI functionality of these packages was moved to `zope.intid` with backwards compatibility imports left in place. Review your imports from `zope.app.intid` to see whether they cannot come directly from `zope.intid` instead.

### zope.app.catalog -> zope.catalog

The non-UI functionality of these packages was moved to `zope.catalog`. Review your imports from `zope.app.catalog` to see whether they cannot come directly from `zope.catalog` instead.

### zope.app.container -> zope.container

The non-UI functionality of these packages was moved to `zope.container`. Review your imports from `zope.app.container` to see whether they cannot come directly from `zope.container` instead.

In addition, the exceptions used by `zope.container` were changed, so if your code catches them, you need to review it:

- The `DuplicationError` in `setitem` was changed to `KeyError`.
- The `UserError` in `NameChooser` was changed to `ValueError`.

### zope.app.component -> zope.security, zope.site

The implementation of the `<class>` ZCML directive moved from this package to `zope.security`. Packages that relied on `zope.app.component` to obtain this directive should declare a direct dependency on `zope.security`, and it may be possible to lose the dependency on `zope.app.component` altogether.

Non-UI site related functionality has been moved to the `zope.site` package. with backwards compatibility imports left in place. Review your imports from `zope.app.component` to see whether they cannot come directly from `zope.site` instead.

### zope.app.folder -> zope.site, zope.container

The implementation of the `zope.app.folder.Folder` class has moved to `zope.site.folder` instead, with backwards compatibility imports left in place. Review your imports from `zope.app.folder` to see whether they cannot come directly from `zope.site` instead. In addition, `Folder` is an `IContainer` implementation that also mixes in site management functionality. If such site management support is not necessary, in some cases your code does not need `Folder` but may be able to rely on a `Container` implementation from `zope.container` instead.

A base class with the implementation of the container-like behavior of `Folder` has moved to `zope.container` (and `zope.site` uses this for its implementation of `Folder`). This is not normally something you should need to retain backwards compatibility.

### zc.copy -> zope.copy, zope.copypastemove, zope.location

The pluggable object copying mechanism once developed in the `zc.copy` package was merged back into `zope.location`, `zope.copypastemove` and the new `zope.copy` package. The `zope.copy` package now provides a pluggable mechanism for copying objects from `zc.copy` and doesn't depend on anything but `zope.interface`. The `zope.copypastemove` uses the `copy` function from `zope.copy` in its `ObjectCopier`.

The zope.location now provides an ICopyHook adapter that implements conditional copy functionality based on object locations, that old zope.location.pickling.CopyPersistent used to provide. Note, that if you don't use ZCML configuration of zope.location, you may need to register zope.location.pickling.LocationCopyHook yourself.

The zope.location.pickling.locationCopy and zope.location.pickling.CopyPersistent are now deprecated in favor of zope.copy and were replaced by backward-compatibility imports. See zope.copy package documentation for information on how to use the new mechanism.

The new version of the zc.copy package now only contains backward-compatibility imports and is deprecated. zope.copy should be preferred for new developments.

### zope.app.security refactoring

The zope.app.security package was finally refactored into a few small parts with less dependencies and more clear purpose.

The implementation of the <module> ZCML directive moved from this package to zope.security. Packages that relied on zope.app.security to obtain this directive should declare a direct dependency on zope.security, and it may be possible to lose the dependency on zope.app.security altogether.

The protectclass module in this package has moved to zope.security, with backwards compatibility imports left in place. Review your imports from zope.app.security to see whether they cannot come directly from zope.security instead.

All interfaces (*IAuthentication*, *IUnauthenticatedPrincipal*, *ILoginPassword* and so on.) were moved into a new zope.authentication package, as well as several utility things, like *PrincipalSource* and *checkPrincipal* function. The new package has much less dependencies and defines an abstract contract for implementing authentication policies. While backward compatibility imports are left in place, it's strongly recommended to update your imports to the zope.authentication.

The *global principal registry* and its ZCML directives are moved into a new zope.principalregistry package with backward-compatibility imports left in place. If your application uses global principals, review your code and ZCML configuration to update it to the new place.

The *local permission* functionality was moved into a new zope.app.localpermission package. This functionality is a part of Through-The-Web development pattern that seems not to be used and supported much by Zope Toolkit and Application anymore, so it can be considered deprecated. However, it can serve as a great example of TTW-related component.

The *Permission vocabularies* and standard protections for Message objects and *__name__*, *__parent__* attributes as well as some common permissions, like *zope.View* and *zope.ManageContent* were merged into *zope.security*.

The adapters from zope.publisher's *IHTTPCredentials* and *IFTPCredentials* to the *ILoginPassword* were moved into zope.publisher, thus making zope.authentication a dependency for zope.publisher.

The original zope.app.security package now only contains several deprecated or application-specific permission definitions, python module protections, that are only likely to be needed with deprecated Through-The-Web development pattern, and ZMI-related browser views (login.html, zope.app.form view for PrincipalSource and so on), as well as backward-compatibility imports. So, if you're not using TTW and/or standard ZMI browser views, you probably should review update your imports to a new places and drop dependency on zope.app.security to reduce package dependencies count.

Other packages, that used zope.app.security, like zope.securitypolicy are either already adapted to the changes or will be adapted soon.

### zope.app.publisher refactoring

The `zope.app.publisher` package was also refactored into smaller parts with less dependencies and clearer purpose.

The browser resources mechanism (mostly used for serving static files and directories) was factored out to the new `zope.browserresource` package. It was also made more pluggable, so you can register specific resource classes for some file extensions, if you need special processing. One of the example is the new `zope.ptresource` package, where the PageTemplateResource was moved, another example is `z3c.zrtresource` package that was adapted to automatically use ZRT resource class for files with `.zrt` extensions.

Browser menu mechanism was moved into a new `zope.browsermenu` package with no further changes.

ZCML directives for easy creation of browser views (the `browser:page` directive and friends) was moved into a new small package, `zope.browserpage`. Also, the directives don't depend the menu mechanism now and will simply ignore "menu" and "title" arguments if `zope.browsermenu` package is not installed.

The `IModifiableBrowserLanguages` adapter was moved into `zope.publisher` along with several ZCML security declarations for `zope.publisher` classes that used to be in `zope.app.publisher`.

ZCML registrations for `IXMLRPCPublisher` adapter for containers was moved into the `zope.container`, because the actual adapters code were already in `zope.container` and registered there as `IBrowserPublisher` adapters. However, both adapters and their ZCML registrations will probably move elsewhere when we'll be refactoring `zope.container`.

Several parts are left in `zope.app.publisher` untouched:

- `Browser Skins` vocabulary.
- `date` field converter for `zope.publisher`'s form values conversion mechanism.
- `ManagementViewSelector` browser view (ZMI-related part).
- `xmlrpc:view` directive for publishing XML-RPC methods.

The latter, `xmlrpc:view` directive is generally useful, so it may be moved into a separate package in future, however there are no clear decision about how to move XML-RPC and FTP-related things currently.

### Password managers extracted from zope.app.authentication

The *IPasswordManager* interface and its implementations were extracted from `zope.app.authentication` into a new `zope.password` package to make them usable with other authentication systems, like `z3c.authenticator` or `zope.principalregistry` or any custom one.

It basically depends only on `zope.interface`, so it can be really useful even in non-Zope environments, like `Pylons`, for example.

The *Password Manager Names* vocabulary is also moved into `zope.password`, however, it's only useful with `zope.schema` and `zope.component`, so you need them installed to work with them. They're listed in the "vocabulary" extra requirement specification.

### ZODB 3.9 FileStorage native blob support

The FileStorage component of ZODB 3.9 used in Zope Toolkit 1.0 now supports blobs natively, so you don't need to use BlobStorage proxy for it anymore.

Thus, you can specify blob directory directly to FileStorage. If you use ZConfig, that means something like this:

```
<filestorage>
  path var/Data.fs
  blob-dir var/blobs
</filestorage>
```

instead of:

```
<blobstorage>
  blob-dir var/blobs
  <filestorage>
    path var/Data.fs
  </filestorage>
</blobstorage>
```

If you creating a storage from python, that means something like this:

```
storage = FileStorage('var/Data.fs', blob_dir='var/blobs')
```

instead of:

```
storage = BlobStorage('var/blobs', FileStorage('var/Data.fs'))
```

### zope.dublincore permission renaming

The `zope.app.dublincore.*` permissions have been renamed to `zope.dublincore.*`. Applications using these permissions have to fix up grants based on the old permissions.

### Zope Toolkit 1.0.3

This document covers major changes in this release that can lead to backward-incompatibilities and explains what to look out for when updating.

### Introduction

The Zope Toolkit 1.0 release is the first release of the Zope Toolkit. The Zope Toolkit really is just a collection of libraries managed together by the Zope developers. We typically treat each library independently, so you would like to look at the CHANGES.txt in each library for updates. Here we note larger changes, especially ones that affect multiple libraries.

### Installation

The Zope Toolkit cannot be installed directly except as individual libraries (such as `zope.component`). To install it you typically would install a framework or application that makes use of these libraries. Examples of such projects are BlueBream, Grok and Zope 2.

If you want to use the Zope Toolkit KGS, you can use the buildout extends mechanism (replace 1.0 by the desired version):

```
[buildout]
extends = http://download.zope.org/zopetoolkit/index/1.0/ztk-versions.cfg
```

You can also copy the file locally or additionally extend the zopeapp-versions.cfg file from the same location.

Frameworks and applications have their own set of install instructions. You should follow these in most cases.

### Python versions

The ZTK 1.0 release series supports Python 2.4 up to Python 2.6. Neither Python 2.7 nor any Python 3.x series is supported.

### News

The 1.0 release of the Zope Toolkit contains a number of refactorings that are aimed to clean up dependencies between pieces of code. Many packages in `zope.app` have had their code moved to existing or newly created packages in the `zope` namespace. These new packages do generally not contain user interface code (typically what's in `.browser`), and have much clearer dependency relationships as a result.

Backwards compatibility imports have been left in place so that your existing code should still work. In some cases you will have to explicitly add dependencies to a `zope.app.` to your code, as due to the cleanup they might not come in automatically anymore due to indirect dependencies; if you see an import error this is probably the case.

We recommend you update your existing code to import from the new packages if possible. The transition support and `zope.app.*` support is limited: the legacy support will be officially retired from the ZTK in subsequent releases.

### Migration issues

#### zope.app.keyreference -> zope.keyreference

This package was renamed to `zope.keyreference` and all its functionality was moved to the new one. The new package contains a little workaround for making old persistent keyreferences loadable without `zope.app.keyreference` installed, so the latter one is not needed at all anymore. Still review your code for any imports coming from `zope.app.keyreference` and modify it to use `zope.keyreference` instead.

#### zope.app.intid -> zope.intid

The non-UI functionality of these packages was moved to `zope.intid` with backwards compatibility imports left in place. Review your imports from `zope.app.intid` to see whether they cannot come directly from `zope.intid` instead.

#### zope.app.catalog -> zope.catalog

The non-UI functionality of these packages was moved to `zope.catalog`. Review your imports from `zope.app.catalog` to see whether they cannot come directly from `zope.catalog` instead.

#### zope.app.container -> zope.container

The non-UI functionality of these packages was moved to `zope.container`. Review your imports from `zope.app.container` to see whether they cannot come directly from `zope.container` instead.

In addition, the exceptions used by `zope.container` were changed, so if your code catches them, you need to review it:

- The `DuplicationError` in `setitem` was changed to `KeyError`.
- The `UserError` in `NameChooser` was changed to `ValueError`.

### zope.app.component -> zope.security, zope.site

The implementation of the `<class>` ZCML directive moved from this package to `zope.security`. Packages that relied on `zope.app.component` to obtain this directive should declare a direct dependency on `zope.security`, and it may be possible to lose the dependency on `zope.app.component` altogether.

Non-UI site related functionality has been moved to the `zope.site` package. with backwards compatibility imports left in place. Review your imports from `zope.app.component` to see whether they cannot come directly from `zope.site` instead.

### zope.app.folder -> zope.site, zope.container

The implementation of the `zope.app.folder.Folder` class has moved to `zope.site.folder` instead, with backwards compatibility imports left in place. Review your imports from `zope.app.folder` to see whether they cannot come directly from `zope.site` instead. In addition, `Folder` is an `IContainer` implementation that also mixes in site management functionality. If such site management support is not necessary, in some cases your code does not need `Folder` but may be able to rely on a `Container` implementation from `zope.container` instead.

A base class with the implementation of the container-like behavior of `Folder` has moved to `zope.container` (and `zope.site` uses this for its implementation of `Folder`). This is not normally something you should need to retain backwards compatibility.

### zc.copy -> zope.copy, zope.copypastemove, zope.location

The pluggable object copying mechanism once developed in the `zc.copy` package was merged back into `zope.location`, `zope.copypastemove` and the new `zope.copy` package. The `zope.copy` package now provides a pluggable mechanism for copying objects from `zc.copy` and doesn't depend on anything but `zope.interface`. The `zope.copypastemove` uses the `copy` function from `zope.copy` in its `ObjectCopier`.

The `zope.location` now provides an `ICopyHook` adapter that implements conditional copy functionality based on object locations, that old `zope.location.pickling.CopyPersistent` used to provide. Note, that if you don't use ZCML configuration of `zope.location`, you may need to register `zope.location.pickling.LocationCopyHook` yourself.

The `zope.location.pickling.locationCopy` and `zope.location.pickling.CopyPersistent` are now deprecated in favor of `zope.copy` and were replaced by backward-compatibility imports. See `zope.copy` package documentation for information on how to use the new mechanism.

The new version of the `zc.copy` package now only contains backward-compatibility imports and is deprecated. `zope.copy` should be preferred for new developments.

### zope.app.security refactoring

The `zope.app.security` package was finally refactored into a few small parts with less dependencies and more clear purpose.

The implementation of the `<module>` ZCML directive moved from this package to `zope.security`. Packages that relied on `zope.app.security` to obtain this directive should declare a direct dependency on `zope.security`, and it may be possible to lose the dependency on `zope.app.security` altogether.

The `protectclass` module in this package has moved to `zope.security`, with backwards compatibility imports left in place. Review your imports from `zope.app.security` to see whether they cannot come directly from `zope.security` instead.

All interfaces (*IAuthentication*, *IUnauthenticatedPrincipal*, *ILoginPassword* and so on.) were moved into a new `zope.authentication` package, as well as several utility things, like *PrincipalSource* and *checkPrincipal* function. The new package has much less dependencies and defines an abstract contract for implementing authentication policies. While backward compatibility imports are left in place, it's strongly recommended to update your imports to the `zope.authentication`.

The *global principal registry* and its ZCML directives are moved into a new `zope.principalregistry` package with backward-compatibility imports left in place. If your application uses global principals, review your code and ZCML configuration to update it to the new place.

The *local permission* functionality was moved into a new `zope.app.localpermission` package. This functionality is a part of Through-The-Web development pattern that seems not to be used and supported much by Zope Toolkit and Application anymore, so it can be considered deprecated. However, it can serve as a great example of TTW-related component.

The *Permission vocabularies* and standard protections for Message objects and *__name__*, *__parent__* attributes as well as some common permissions, like *zope.View* and *zope.ManageContent* were merged into *zope.security*.

The adapters from `zope.publisher`'s *IHTTPCredentials* and *IFTPCredentials* to the *ILoginPassword* were moved into `zope.publisher`, thus making `zope.authentication` a dependency for `zope.publisher`.

The original `zope.app.security` package now only contains several deprecated or application-specific permission definitions, python module protections, that are only likely to be needed with deprecated Through-The-Web development pattern, and ZMI-related browser views (login.html, zope.app.form view for PrincipalSource and so on), as well as backward-compatibility imports. So, if you're not using TTW and/or standard ZMI browser views, you probably should review update your imports to a new places and drop dependency on `zope.app.security` to reduce package dependencies count.

Other packages, that used `zope.app.security`, like `zope.securitypolicy` are either already adapted to the changes or will be adapted soon.

### zope.app.publisher refactoring

The `zope.app.publisher` package was also refactored into smaller parts with less dependencies and clearer purpose.

The browser resources mechanism (mostly used for serving static files and directories) was factored out to the new `zope.browserresource` package. It was also made more pluggable, so you can register specific resource classes for some file extensions, if you need special processing. One of the example is the new `zope.ptresource` package, where the PageTemplateResource was moved, another example is `z3c.zrtresource` package that was adapted to automatically use ZRT resource class for files with `.zrt` extensions.

Browser menu mechanism was moved into a new `zope.browsermenu` package with no further changes.

ZCML directives for easy creation of browser views (the `browser:page` directive and friends) was moved into a new small package, `zope.browserpage`. Also, the directives don't depend the menu mechanism now and will simply ignore "menu" and "title" arguments if `zope.browsermenu` package is not installed.

The `IModifiableBrowserLanguages` adapter was moved into `zope.publisher` along with several ZCML security declarations for `zope.publisher` classes that used to be in `zope.app.publisher`.

ZCML registrations for `IXMLRPCPublisher` adapter for containers was moved into the `zope.container`, because the actual adapters code were already in `zope.container` and registered there as `IBrowserPublisher` adapters. However, both adapters and their ZCML registrations will probably move elsewhere when we'll be refactoring `zope.container`.

Several parts are left in `zope.app.publisher` untouched:

- `Browser Skins` vocabulary.
- `date` field converter for `zope.publisher`'s form values conversion mechanism.
- `ManagementViewSelector` browser view (ZMI-related part).
- `xmlrpc:view` directive for publishing XML-RPC methods.

The latter, `xmlrpc:view` directive is generally useful, so it may be moved into a separate package in future, however there are no clear decision about how to move XML-RPC and FTP-related things currently.

### Password managers extracted from zope.app.authentication

The *IPasswordManager* interface and its implementations were extracted from `zope.app.authentication` into a new `zope.password` package to make them usable with other authentication systems, like `z3c.authenticator` or `zope.principalregistry` or any custom one.

It basically depends only on `zope.interface`, so it can be really useful even in non-Zope environments, like `Pylons`, for example.

The *Password Manager Names* vocabulary is also moved into `zope.password`, however, it's only useful with `zope.schema` and `zope.component`, so you need them installed to work with them. They're listed in the "vocabulary" extra requirement specification.

### ZODB 3.9 FileStorage native blob support

The FileStorage component of ZODB 3.9 used in Zope Toolkit 1.0 now supports blobs natively, so you don't need to use BlobStorage proxy for it anymore.

Thus, you can specify blob directory directly to FileStorage. If you use ZConfig, that means something like this:

```
<filestorage>
  path var/Data.fs
  blob-dir var/blobs
</filestorage>
```

instead of:

```
<blobstorage>
  blob-dir var/blobs
  <filestorage>
    path var/Data.fs
  </filestorage>
</blobstorage>
```

If you creating a storage from python, that means something like this:

```
storage = FileStorage('var/Data.fs', blob_dir='var/blobs')
```

instead of:

```
storage = BlobStorage('var/blobs', FileStorage('var/Data.fs'))
```

### zope.dublincore permission renaming

The `zope.app.dublincore.*` permissions have been renamed to `zope.dublincore.*`. Applications using these permissions have to fix up grants based on the old permissions.

### Zope Toolkit 1.0.2

This document covers major changes in this release that can lead to backward-incompatibilities and explains what to look out for when updating.

### Introduction

The Zope Toolkit 1.0 release is the first release of the Zope Toolkit. The Zope Toolkit really is just a collection of libraries managed together by the Zope developers. We typically treat each library independently, so you would like to look at the CHANGES.txt in each library for updates. Here we note larger changes, especially ones that affect multiple libraries.

### Installation

The Zope Toolkit cannot be installed directly except as individual libraries (such as `zope.component`). To install it you typically would install a framework or application that makes use of these libraries. Examples of such projects are BlueBream, Grok and Zope 2.

If you want to use the Zope Toolkit KGS, you can use the buildout extends mechanism (replace 1.0 by the desired version):

```
[buildout]
extends = http://download.zope.org/zopetoolkit/index/1.0/ztk-versions.cfg
```

You can also copy the file locally or additionally extend the zopeapp-versions.cfg file from the same location.

Frameworks and applications have their own set of install instructions. You should follow these in most cases.

### Python versions

The ZTK 1.0 release series supports Python 2.4 up to Python 2.6. Neither Python 2.7 nor any Python 3.x series is supported.

### News

The 1.0 release of the Zope Toolkit contains a number of refactorings that are aimed to clean up dependencies between pieces of code. Many packages in `zope.app` have had their code moved to existing or newly created packages in the `zope` namespace. These new packages do generally not contain user interface code (typically what's in `.browser`), and have much clearer dependency relationships as a result.

Backwards compatibility imports have been left in place so that your existing code should still work. In some cases you will have to explicitly add dependencies to a `zope.app.` to your code, as due to the cleanup they might not come in automatically anymore due to indirect dependencies; if you see an import error this is probably the case.

We recommend you update your existing code to import from the new packages if possible. The transition support and `zope.app.*` support is limited: the legacy support will be officially retired from the ZTK in subsequent releases.

### Migration issues

#### zope.app.keyreference -> zope.keyreference

This package was renamed to `zope.keyreference` and all its functionality was moved to the new one. The new package contains a little workaround for making old persistent keyreferences loadable without `zope.app.keyreference` installed, so the latter one is not needed at all anymore. Still review your code for any imports coming from `zope.app.keyreference` and modify it to use `zope.keyreference` instead.

#### zope.app.intid -> zope.intid

The non-UI functionality of these packages was moved to `zope.intid` with backwards compatibility imports left in place. Review your imports from `zope.app.intid` to see whether they cannot come directly from `zope.intid` instead.

#### zope.app.catalog -> zope.catalog

The non-UI functionality of these packages was moved to `zope.catalog`. Review your imports from `zope.app.catalog` to see whether they cannot come directly from `zope.catalog` instead.

#### zope.app.container -> zope.container

The non-UI functionality of these packages was moved to `zope.container`. Review your imports from `zope.app.container` to see whether they cannot come directly from `zope.container` instead.

In addition, the exceptions used by `zope.container` were changed, so if your code catches them, you need to review it:

- The `DuplicationError` in `setitem` was changed to `KeyError`.
- The `UserError` in `NameChooser` was changed to `ValueError`.

#### zope.app.component -> zope.security, zope.site

The implementation of the `<class>` ZCML directive moved from this package to `zope.security`. Packages that relied on `zope.app.component` to obtain this directive should declare a direct dependency on `zope.security`, and it may be possible to lose the dependency on `zope.app.component` altogether.

Non-UI site related functionality has been moved to the `zope.site` package. with backwards compatibility imports left in place. Review your imports from `zope.app.component` to see whether they cannot come directly from `zope.site` instead.

#### zope.app.folder -> zope.site, zope.container

The implementation of the `zope.app.folder.Folder` class has moved to `zope.site.folder` instead, with backwards compatibility imports left in place. Review your imports from `zope.app.folder` to see whether they cannot come directly from `zope.site` instead. In addition, `Folder` is an `IContainer` implementation that also mixes in site management functionality. If such site management support is not necessary, in some cases your code does not need `Folder` but may be able to rely on a `Container` implementation from `zope.container` instead.

A base class with the implementation of the container-like behavior of `Folder` has moved to `zope.container` (and `zope.site` uses this for its implementation of `Folder`). This is not normally something you should need to retain backwards compatibility.

### zc.copy -> zope.copy, zope.copypastemove, zope.location

The pluggable object copying mechanism once developed in the `zc.copy` package was merged back into `zope.location`, `zope.copypastemove` and the new `zope.copy` package. The `zope.copy` package now provides a pluggable mechanism for copying objects from `zc.copy` and doesn't depend on anything but `zope.interface`. The `zope.copypastemove` uses the `copy` function from `zope.copy` in its `ObjectCopier`.

The `zope.location` now provides an `ICopyHook` adapter that implements conditional copy functionality based on object locations, that old `zope.location.pickling.CopyPersistent` used to provide. Note, that if you don't use ZCML configuration of `zope.location`, you may need to register `zope.location.pickling.LocationCopyHook` yourself.

The `zope.location.pickling.locationCopy` and `zope.location.pickling.CopyPersistent` are now deprecated in favor of `zope.copy` and were replaced by backward-compatibility imports. See `zope.copy` package documentation for information on how to use the new mechanism.

The new version of the `zc.copy` package now only contains backward-compatibility imports and is deprecated. `zope.copy` should be preferred for new developments.

### zope.app.security refactoring

The `zope.app.security` package was finally refactored into a few small parts with less dependencies and more clear purpose.

The implementation of the `<module>` ZCML directive moved from this package to `zope.security`. Packages that relied on `zope.app.security` to obtain this directive should declare a direct dependency on `zope.security`, and it may be possible to lose the dependency on `zope.app.security` altogether.

The `protectclass` module in this package has moved to `zope.security`, with backwards compatibility imports left in place. Review your imports from `zope.app.security` to see whether they cannot come directly from `zope.security` instead.

All interfaces (*IAuthentication*, *IUnauthenticatedPrincipal*, *ILoginPassword* and so on.) were moved into a new `zope.authentication` package, as well as several utility things, like *PrincipalSource* and *checkPrincipal* function. The new package has much less dependencies and defines an abstract contract for implementing authentication policies. While backward compatibility imports are left in place, it's strongly recommended to update your imports to the `zope.authentication`.

The *global principal registry* and its ZCML directives are moved into a new `zope.principalregistry` package with backward-compatibility imports left in place. If your application uses global principals, review your code and ZCML configuration to update it to the new place.

The *local permission* functionality was moved into a new `zope.app.localpermission` package. This functionality is a part of Through-The-Web development pattern that seems not to be used and supported much by Zope Toolkit and Application anymore, so it can be considered deprecated. However, it can serve as a great example of TTW-related component.

The *Permission vocabularies* and standard protections for Message objects and *__name__*, *__parent__* attributes as well as some common permissions, like *zope.View* and *zope.ManageContent* were merged into *zope.security*.

The adapters from `zope.publisher`'s *IHTTPCredentials* and *IFTPCredentials* to the *ILoginPassword* were moved into `zope.publisher`, thus making `zope.authentication` a dependency for `zope.publisher`.

The original zope.app.security package now only contains several deprecated or application-specific permission definitions, python module protections, that are only likely to be needed with deprecated Through-The-Web development pattern, and ZMI-related browser views (login.html, zope.app.form view for PrincipalSource and so on), as well as backward-compatibility imports. So, if you're not using TTW and/or standard ZMI browser views, you probably should review update your imports to a new places and drop dependency on zope.app.security to reduce package dependencies count.

Other packages, that used zope.app.security, like zope.securitypolicy are either already adapted to the changes or will be adapted soon.

### zope.app.publisher refactoring

The zope.app.publisher package was also refactored into smaller parts with less dependencies and clearer purpose.

The browser resources mechanism (mostly used for serving static files and directories) was factored out to the new zope.browserresource package. It was also made more pluggable, so you can register specific resource classes for some file extensions, if you need special processing. One of the example is the new zope.ptresource package, where the PageTemplateResource was moved, another example is z3c.zrtresource package that was adapted to automatically use ZRT resource class for files with .zrt extensions.

Browser menu mechanism was moved into a new zope.browsermenu package with no further changes.

ZCML directives for easy creation of browser views (the browser:page directive and friends) was moved into a new small package, zope.browserpage. Also, the directives don't depend the menu mechanism now and will simply ignore "menu" and "title" arguments if zope.browsermenu package is not installed.

The IModifiableBrowserLanguages adapter was moved into zope.publisher along with several ZCML security declarations for zope.publisher classes that used to be in zope.app.publisher.

ZCML registrations for IXMLRPCPublisher adapter for containers was moved into the zope.container, because the actual adapters code were already in zope.container and registered there as IBrowserPublisher adapters. However, both adapters and their ZCML registrations will probably move elsewhere when we'll be refactoring zope.container.

Several parts are left in zope.app.publisher untouched:

- Browser Skins vocabulary.
- date field converter for zope.publisher's form values conversion mechanism.
- ManagementViewSelector browser view (ZMI-related part).
- xmlrpc:view directive for publishing XML-RPC methods.

The latter, xmlrpc:view directive is generally useful, so it may be moved into a separate package in future, however there are no clear decision about how to move XML-RPC and FTP-related things currently.

### Password managers extracted from zope.app.authentication

The *IPasswordManager* interface and its implementations were extracted from zope.app.authentication into a new zope.password package to make them usable with other authentication systems, like z3c.authenticator or zope.principalregistry or any custom one.

It basically depends only on zope.interface, so it can be really useful even in non-Zope environments, like Pylons, for example.

The *Password Manager Names* vocabulary is also moved into `zope.password`, however, it's only useful with `zope.schema` and `zope.component`, so you need them installed to work with them. They're listed in the "vocabulary" extra requirement specification.

### ZODB 3.9 FileStorage native blob support

The FileStorage component of ZODB 3.9 used in Zope Toolkit 1.0 now supports blobs natively, so you don't need to use BlobStorage proxy for it anymore.

Thus, you can specify blob directory directly to FileStorage. If you use ZConfig, that means something like this:

```
<filestorage>
  path var/Data.fs
  blob-dir var/blobs
</filestorage>
```

instead of:

```
<blobstorage>
  blob-dir var/blobs
  <filestorage>
    path var/Data.fs
  </filestorage>
</blobstorage>
```

If you creating a storage from python, that means something like this:

```
storage = FileStorage('var/Data.fs', blob_dir='var/blobs')
```

instead of:

```
storage = BlobStorage('var/blobs', FileStorage('var/Data.fs'))
```

### zope.dublincore permission renaming

The `zope.app.dublincore.*` permissions have been renamed to `zope.dublincore.*`. Applications using these permissions have to fix up grants based on the old permissions.

### Zope Toolkit 1.0.1

This document covers major changes in this release that can lead to backward-incompatibilities and explains what to look out for when updating.

### Introduction

The Zope Toolkit 1.0 release is the first release of the Zope Toolkit. The Zope Toolkit really is just a collection of libraries managed together by the Zope developers. We typically treat each library independently, so you would like to look at the CHANGES.txt in each library for updates. Here we note larger changes, especially ones that affect multiple libraries.

### Installation

The Zope Toolkit cannot be installed directly except as individual libraries (such as `zope.component`). To install it you typically would install a framework or application that makes use of these libraries. Examples of such projects are BlueBream, Grok and Zope 2.

If you want to use the Zope Toolkit KGS, you can use the buildout extends mechanism (replace 1.0 by the desired version):

```
[buildout]
extends = http://download.zope.org/zopetoolkit/index/1.0/ztk-versions.cfg
```

You can also copy the file locally or additionally extend the zopeapp-versions.cfg file from the same location.

Frameworks and applications have their own set of install instructions. You should follow these in most cases.

### News

The 1.0 release of the Zope Toolkit contains a number of refactorings that are aimed to clean up dependencies between pieces of code. Many packages in `zope.app` have had their code moved to existing or newly created packages in the `zope` namespace. These new packages do generally not contain user interface code (typically what's in `.browser`), and have much clearer dependency relationships as a result.

Backwards compatibility imports have been left in place so that your existing code should still work. In some cases you will have to explicitly add dependencies to a `zope.app.` to your code, as due to the cleanup they might not come in automatically anymore due to indirect dependencies; if you see an import error this is probably the case.

We recommend you update your existing code to import from the new packages if possible. The transition support and `zope.app.*` support is limited: the legacy support will be officially retired from the ZTK after January 1, 2011.

### Migration issues

#### zope.app.keyreference -> zope.keyreference

This package was renamed to `zope.keyreference` and all its functionality was moved to the new one. The new package contains a little workaround for making old persistent keyreferences loadable without `zope.app.keyreference` installed, so the latter one is not needed at all anymore. Still review your code for any imports coming from `zope.app.keyreference` and modify it to use `zope.keyreference` instead.

#### zope.app.intid -> zope.intid

The non-UI functionality of these packages was moved to `zope.intid` with backwards compatibility imports left in place. Review your imports from `zope.app.intid` to see whether they cannot come directly from `zope.intid` instead.

#### zope.app.catalog -> zope.catalog

The non-UI functionality of these packages was moved to `zope.catalog`. Review your imports from `zope.app.catalog` to see whether they cannot come directly from `zope.catalog` instead.

### zope.app.container -> zope.container

The non-UI functionality of these packages was moved to `zope.container`. Review your imports from `zope.app.container` to see whether they cannot come directly from `zope.container` instead.

In addition, the exceptions used by `zope.container` were changed, so if your code catches them, you need to review it:

- The `DuplicationError` in `setitem` was changed to `KeyError`.
- The `UserError` in `NameChooser` was changed to `ValueError`.

### zope.app.component -> zope.security, zope.site

The implementation of the `<class>` ZCML directive moved from this package to `zope.security`. Packages that relied on `zope.app.component` to obtain this directive should declare a direct dependency on `zope.security`, and it may be possible to lose the dependency on `zope.app.component` altogether.

Non-UI site related functionality has been moved to the `zope.site` package. with backwards compatibility imports left in place. Review your imports from `zope.app.component` to see whether they cannot come directly from `zope.site` instead.

### zope.app.folder -> zope.site, zope.container

The implementation of the `zope.app.folder.Folder` class has moved to `zope.site.folder` instead, with backwards compatibility imports left in place. Review your imports from `zope.app.folder` to see whether they cannot come directly from `zope.site` instead. In addition, `Folder` is an `IContainer` implementation that also mixes in site management functionality. If such site management support is not necessary, in some cases your code does not need `Folder` but may be able to rely on a `Container` implementation from `zope.container` instead.

A base class with the implementation of the container-like behavior of `Folder` has moved to `zope.container` (and `zope.site` uses this for its implementation of `Folder`). This is not normally something you should need to retain backwards compatibility.

### zc.copy -> zope.copy, zope.copypastemove, zope.location

The pluggable object copying mechanism once developed in the `zc.copy` package was merged back into `zope.location`, `zope.copypastemove` and the new `zope.copy` package. The `zope.copy` package now provides a pluggable mechanism for copying objects from `zc.copy` and doesn't depend on anything but `zope.interface`. The `zope.copypastemove` uses the `copy` function from `zope.copy` in its `ObjectCopier`.

The `zope.location` now provides an `ICopyHook` adapter that implements conditional copy functionality based on object locations, that old `zope.location.pickling.CopyPersistent` used to provide. Note, that if you don't use ZCML configuration of `zope.location`, you may need to register `zope.location.pickling.LocationCopyHook` yourself.

The `zope.location.pickling.locationCopy` and `zope.location.pickling.CopyPersistent` are now deprecated in favor of `zope.copy` and were replaced by backward-compatibility imports. See `zope.copy` package documentation for information on how to use the new mechanism.

The new version of the `zc.copy` package now only contains backward-compatibility imports and is deprecated. `zope.copy` should be preferred for new developments.

### zope.app.security refactoring

The `zope.app.security` package was finally refactored into a few small parts with less dependencies and more clear purpose.

The implementation of the `<module>` ZCML directive moved from this package to `zope.security`. Packages that relied on `zope.app.security` to obtain this directive should declare a direct dependency on `zope.security`, and it may be possible to lose the dependency on `zope.app.security` altogether.

The `protectclass` module in this package has moved to `zope.security`, with backwards compatibility imports left in place. Review your imports from `zope.app.security` to see whether they cannot come directly from `zope.security` instead.

All interfaces (*IAuthentication*, *IUnauthenticatedPrincipal*, *ILoginPassword* and so on.) were moved into a new `zope.authentication` package, as well as several utility things, like *PrincipalSource* and *checkPrincipal* function. The new package has much less dependencies and defines an abstract contract for implementing authentication policies. While backward compatibility imports are left in place, it's strongly recommended to update your imports to the `zope.authentication`.

The *global principal registry* and its ZCML directives are moved into a new `zope.principalregistry` package with backward-compatibility imports left in place. If your application uses global principals, review your code and ZCML configuration to update it to the new place.

The *local permission* functionality was moved into a new `zope.app.localpermission` package. This functionality is a part of Through-The-Web development pattern that seems not to be used and supported much by Zope Toolkit and Application anymore, so it can be considered deprecated. However, it can serve as a great example of TTW-related component.

The *Permission vocabularies* and standard protections for Message objects and *__name__*, *__parent__* attributes as well as some common permissions, like *zope.View* and *zope.ManageContent* were merged into *zope.security*.

The adapters from `zope.publisher`'s *IHTTPCredentials* and *IFTPCredentials* to the *ILoginPassword* were moved into `zope.publisher`, thus making `zope.authentication` a dependency for `zope.publisher`.

The original `zope.app.security` package now only contains several deprecated or application-specific permission definitions, python module protections, that are only likely to be needed with deprecated Through-The-Web development pattern, and ZMI-related browser views (login.html, zope.app.form view for PrincipalSource and so on), as well as backward-compatibility imports. So, if you're not using TTW and/or standard ZMI browser views, you probably should review update your imports to a new places and drop dependency on `zope.app.security` to reduce package dependencies count.

Other packages, that used `zope.app.security`, like `zope.securitypolicy` are either already adapted to the changes or will be adapted soon.

### zope.app.publisher refactoring

The `zope.app.publisher` package was also refactored into smaller parts with less dependencies and clearer purpose.

The browser resources mechanism (mostly used for serving static files and directories) was factored out to the new `zope.browserresource` package. It was also made more pluggable, so you can register specific resource classes for some file extensions, if you need special processing. One of the example is the new `zope.ptresource` package, where the PageTemplateResource was moved, another example is `z3c.zrtresource` package that was adapted to automatically use ZRT resource class for files with `.zrt` extensions.

Browser menu mechanism was moved into a new `zope.browsermenu` package with no further changes.

ZCML directives for easy creation of browser views (the `browser:page` directive and friends) was moved into a new small package, `zope.browserpage`. Also, the directives don't depend the menu mechanism now and will simply ignore "menu" and "title" arguments if `zope.browsermenu` package is not installed.

The `IModifiableBrowserLanguages` adapter was moved into `zope.publisher` along with several ZCML security declarations for `zope.publisher` classes that used to be in `zope.app.publisher`.

ZCML registrations for `IXMLRPCPublisher` adapter for containers was moved into the `zope.container`, because the actual adapters code were already in `zope.container` and registered there as `IBrowserPublisher` adapters. However, both adapters and their ZCML registrations will probably move elsewhere when we'll be refactoring `zope.container`.

Several parts are left in `zope.app.publisher` untouched:

- `Browser Skins` vocabulary.
- `date` field converter for `zope.publisher`'s form values conversion mechanism.
- `ManagementViewSelector` browser view (ZMI-related part).
- `xmlrpc:view` directive for publishing XML-RPC methods.

The latter, `xmlrpc:view` directive is generally useful, so it may be moved into a separate package in future, however there are no clear decision about how to move XML-RPC and FTP-related things currently.

### Password managers extracted from zope.app.authentication

The *IPasswordManager* interface and its implementations were extracted from `zope.app.authentication` into a new `zope.password` package to make them usable with other authentication systems, like `z3c.authenticator` or `zope.principalregistry` or any custom one.

It basically depends only on `zope.interface`, so it can be really useful even in non-Zope environments, like `Pylons`, for example.

The *Password Manager Names* vocabulary is also moved into `zope.password`, however, it's only useful with `zope.schema` and `zope.component`, so you need them installed to work with them. They're listed in the "vocabulary" extra requirement specification.

### ZODB 3.9 FileStorage native blob support

The FileStorage component of ZODB 3.9 used in Zope Toolkit 1.0 now supports blobs natively, so you don't need to use BlobStorage proxy for it anymore.

Thus, you can specify blob directory directly to FileStorage. If you use ZConfig, that means something like this:

```
<filestorage>
  path var/Data.fs
  blob-dir var/blobs
</filestorage>
```

instead of:

```
<blobstorage>
  blob-dir var/blobs
  <filestorage>
    path var/Data.fs
  </filestorage>
</blobstorage>
```

If you creating a storage from python, that means something like this:

```
storage = FileStorage('var/Data.fs', blob_dir='var/blobs')
```

instead of:

```
storage = BlobStorage('var/blobs', FileStorage('var/Data.fs'))
```

#### zope.dublincore permission renaming

The `zope.app.dublincore.*` permissions have been renamed to `zope.dublincore.*`. Applications using these permissions have to fix up grants based on the old permissions.

## Weekly Zope developer meetings

### 2010-11-16

This is the agenda and summary for the weekly Zope developer meeting of Tuesday, 2010-11-16 on #zope@irc.freenode.org from 15:00 to 15:30 UTC.

#### Summary

**The IRC log is available here:** XXX

#### Attendees

XXX

#### Agenda

- PloneConf reports? Grok sprint reports?
- Repository policy nagging script?
- Follow-up on summit goals
- Bicycle toolkit

#### Summary

Nothing happened as Christian and Marius were the only attends and both both timing issues with DST were suspected (nobody should up after the time, though) and also people might have been exhausted from the Plone Conference the week before.

### Ongoing issues

Those issues are currently ongoing. We don't have to discuss them. We just need to follow up on them eventually.

- **Meta**
  - Review meeting itself, maybe add extra 15 minutes for "meta" once a month or every two months? (postponed until 2010-06-01)
  - How to organize open issues in the long run (Blueprints? Other tool? Continue text files?)
- Repository policy
- **Test runners / nightly builds**
  - **Supporting Python 2.7**
    - Needs help from the buildbots
  - **Windows**
    - Compiler licenses (Tres, postponed until after 2010-06-14)
  - Build bot organization
- Bug day organization
- **Documentation**
  - **Consolidate "floating" documentation into Sphinx/docs.zope.org**
    - Write blueprint for the consolidation effort (Theuni) see https://blueprints.edge.launchpad.net/zopetoolkit-project/+spec/consolidate-documentation
    - Find candidate links and gather them centrally
    - Edit/update the documentation from the link list and land in Sphinx-style during a sprint
  - **Turn ZTK package documentation into sphinx style (like zope.event)**
    - write bug and assign to toolkit projects (Theuni)
  - Unified index?
- **Releases**
  - How to find a good point when to cut a new release for a package for which fixed bugs where registered (or changes have been made)? Any automation possible to alert us when changes have been sitting around unreleased for a while?

### Topic proposals

- Chris McDonough: Pondering *some* (re-)structuring of the ZTK to allow for better maintenance/release management/communication/marketing.
- Christian Theune: I'd like us to ponder how we can (in addition to the housekeeping and cleanups we do) also move to do constructive work together to expand the stuff that Zope packages (ZTK) is about. How do we go about implementing new technologies together, like supporting HTML 5 in the various parts? I'd like to start putting in new code in the foreseeable future in the zope.* namespace.

### 2010-11-02

This is the agenda and summary for the weekly Zope developer meeting of Tuesday, 2010-11-02 on #zope@irc.freenode.org from 15:00 to 15:30 UTC.

### Summary

**The IRC log is available here:** http://zope3.pov.lt/irclogs-zope/%23zope.2010-11-02.log.html#t2010-11-02T17:00:49

### Attendees

Christian Theune, Marius Gedminas

### Agenda

- PloneConf reports?
- Follow-up on summit goals
- Bicycle toolkit

### Summary

Nothing happened as Christian and Marius were the only attends and both both timing issues with DST were suspected (nobody should up after the time, though) and also people might have been exhausted from the Plone Conference the week before.

### Ongoing issues

Those issues are currently ongoing. We don't have to discuss them. We just need to follow up on them eventually.

- **Meta**
  - Review meeting itself, maybe add extra 15 minutes for "meta" once a month or every two months? (postponed until 2010-06-01)
  - How to organize open issues in the long run (Blueprints? Other tool? Continue text files?)
- Repository policy
- **Test runners / nightly builds**
  - **Supporting Python 2.7**
    - Needs help from the buildbots
  - **Windows**
    - Compiler licenses (Tres, postponed until after 2010-06-14)
  - Build bot organization
- Bug day organization
- **Documentation**

  - **Consolidate "floating" documentation into Sphinx/docs.zope.org**
    * Write blueprint for the consolidation effort (Theuni) see https://blueprints.edge.launchpad.net/zopetoolkit-project/+spec/consolidate-documentation
    * Find candidate links and gather them centrally
    * Edit/update the documentation from the link list and land in Sphinx-style during a sprint
  - **Turn ZTK package documentation into sphinx style (like zope.event)**
    * write bug and assign to toolkit projects (Theuni)
  - Unified index?

- **Releases**
  - How to find a good point when to cut a new release for a package for which fixed bugs where registered (or changes have been made)? Any automation possible to alert us when changes have been sitting around unreleased for a while?

### Topic proposals

- Chris McDonough: Pondering *some* (re-)structuring of the ZTK to allow for better maintenance/release management/communication/marketing.
- Christian Theune: I'd like us to ponder how we can (in addition to the housekeeping and cleanups we do) also move to do constructive work together to expand the stuff that Zope packages (ZTK) is about. How do we go about implementing new technologies together, like supporting HTML 5 in the various parts? I'd like to start putting in new code in the foreseeable future in the zope.* namespace.

### 2010-10-26

This is the agenda and summary for the weekly Zope developer meeting of Tuesday, 2010-10-26 on #zope@irc.freenode.org from 15:00 to 15:30 UTC.

### Summary

**The IRC log is available here:** http://zope3.pov.lt/irclogs-zope/%23zope.2010-10-26.log.html#t2010-10-26T18:07:26

### Attendees

Christian Theune

### Agenda

- Follow-up on summit goals
- Bicycle toolkit

### Summary

Nothing happened as Christian was the only attendend.

### Ongoing issues

Those issues are currently ongoing. We don't have to discuss them. We just need to follow up on them eventually.

- **Meta**
  - Review meeting itself, maybe add extra 15 minutes for "meta" once a month or every two months? (postponed until 2010-06-01)
  - How to organize open issues in the long run (Blueprints? Other tool? Continue text files?)
- Repository policy
- **Test runners / nightly builds**
  - **Supporting Python 2.7**
    - Needs help from the buildbots
  - **Windows**
    - Compiler licenses (Tres, postponed until after 2010-06-14)
  - Build bot organization
- Bug day organization
- **Documentation**
  - **Consolidate "floating" documentation into Sphinx/docs.zope.org**
    - Write blueprint for the consolidation effort (Theuni) see https://blueprints.edge.launchpad.net/zopetoolkit-project/+spec/consolidate-documentation
    - Find candidate links and gather them centrally
    - Edit/update the documentation from the link list and land in Sphinx-style during a sprint
  - **Turn ZTK package documentation into sphinx style (like zope.event)**
    - write bug and assign to toolkit projects (Theuni)
  - Unified index?
- **Releases**
  - How to find a good point when to cut a new release for a package for which fixed bugs where registered (or changes have been made)? Any automation possible to alert us when changes have been sitting around unreleased for a while?

### Topic proposals

- Chris McDonough: Pondering *some* (re-)structuring of the ZTK to allow for better maintenance/release management/communication/marketing.
- Christian Theune: I'd like us to ponder how we can (in addition to the housekeeping and cleanups we do) also move to do constructive work together to expand the stuff that Zope packages (ZTK) is about. How do we go about implementing new technologies together, like supporting HTML 5 in the various parts? I'd like to start putting in new code in the foreseeable future in the zope.* namespace.

### 2010-10-19

This is the agenda and summary for the weekly Zope developer meeting of Tuesday, 2010-10-19 on #zope@irc.freenode.org from 15:00 to 15:30 UTC.

### Summary

**The IRC log is available here:** http://zope3.pov.lt/irclogs-zope/%23zope.2010-10-19.log.html

### Attendees

Christian Theune, Tres Seaver, Jan-Wijbrandt Kolman, Jim Fulton, Chris McDonough

### Agenda

- Follow-up on summit goals
- Plan next bug day
- Bicycle toolkit

### Follow-up on summit goals

Didn't talk much. Thomas started a thread on discussing the functional areas, but he's sick currently.

### Bug-day

We fixed the next bug day to Tuesday 2010-10-26. Christian promised to declare what he wants to work on in an announcement mail later today to make the bug day activities more visible.

### Bicycle toolkit

Christian mentioned that he finds the BTK seems to overlap with the goal to define functional areas. The BTK might be one of the functional areas. Thomas described one of the possible areas as "software architecture" - this might be identical with the BTK.

### Architecture/design documentation

While talking about the bicycle toolkit a small discussion came up. Jim noted that he'd like a "sane way to talk about architecture" which caused multiple ideas and opinions to be stated.

The basic desire is to have an architectural process/expression that provides enlightenment about design decisions.

The points stated included:

- some architectures are too big to express in a conversation or keep in your head, so an oral tradition breaks down
- people seem to understand an architecture better when it is written in a domain they understand
- having experience with a piece of software makes it easier to understand existing designs in comparison to just reading about it

- the documentation may be useful to reason about and review things like data structures used, problems like index accumulation, and unnecessary indirection
- patterns may include parts of the solution, but specifically the community around patterns seems to be focused on the wrong issues (jargon, blowing smoke)
- recording decisions and being able to relate and search problems and solutions maybe be part of the puzzle

### Ongoing issues

Those issues are currently ongoing. We don't have to discuss them. We just need to follow up on them eventually.

- **Meta**
  - Review meeting itself, maybe add extra 15 minutes for "meta" once a month or every two months? (postponed until 2010-06-01)
  - How to organize open issues in the long run (Blueprints? Other tool? Continue text files?)
- Repository policy
- **Test runners / nightly builds**
  - **Supporting Python 2.7**
    - Needs help from the buildbots
  - **Windows**
    - Compiler licenses (Tres, postponed until after 2010-06-14)
  - Build bot organization
- Bug day organization
- **Documentation**
  - **Consolidate "floating" documentation into Sphinx/docs.zope.org**
    - Write blueprint for the consolidation effort (Theuni) see https://blueprints.edge.launchpad.net/zopetoolkit-project/+spec/consolidate-documentation
    - Find candidate links and gather them centrally
    - Edit/update the documentation from the link list and land in Sphinx-style during a sprint
  - **Turn ZTK package documentation into sphinx style (like zope.event)**
    - write bug and assign to toolkit projects (Theuni)
  - Unified index?
- **Releases**
  - How to find a good point when to cut a new release for a package for which fixed bugs where registered (or changes have been made)? Any automation possible to alert us when changes have been sitting around unreleased for a while?

### Topic proposals

- Chris McDonough: Pondering *some* (re-)structuring of the ZTK to allow for better maintenance/release management/communication/marketing.

- Christian Theune: I'd like us to ponder how we can (in addition to the housekeeping and cleanups we do) also move to do constructive work together to expand the stuff that Zope packages (ZTK) is about. How do we go about implementing new technologies together, like supporting HTML 5 in the various parts? I'd like to start putting in new code in the foreseeable future in the zope.* namespace.

### 2010-10-12

This is the agenda and summary for the weekly Zope developer meeting of Tuesday, 2010-10-12 on #zope@irc.freenode.org from 15:00 to 15:30 UTC.

### Summary

**The IRC log is available here:** http://zope3.pov.lt/irclogs-zope/%23zope.2010-10-12.log.html#t2010-10-12T18:01:32

### Attendees

Christian Theune, Charlie Clark, Adam Groszer, Marius Gedminas

### Summit goal followup

Charlie noted that the time scales on the summit goals are still missing. Christian reported that he originally wanted to gather detail descriptions with everything that was mentioned during the meeting from every individual on the spot but hadn't had a chance to do that. Christian signed up to ask the individual owners of the tasks to fill the details into the wiki (http://wiki.zope.org/ztk/ZopeSummit2010Summary).

As some task owners were present at the meeting, we went through their tasks:

Christian signed up for the Zope sprints. He detailed that he'd like to find topics and interested people for a few dedicated Zope sprints in 2011 until the end of the year. He also keeps an eye on the organization of broader sprints a PyCon and EuroPython as the ZF wanted to push those as well.

Marius is signed up to fix the 5 most incomprehensible error messages by the end of the year. His research is ongoing - he's trying to get people to report those errors to him.

Charlie is signed up to establish/improve a/the developer guide by the end of 2010.

### Bug day

Responses to the current bug day doodle are missing. The last bug days had usually quite a few people subscribe in the Doodle but substantially less showed up.

Apart from unpredictability of schedule or keeping up committments that conflict with business Charlie recommended to make it more clear that subscribing for the whole day is not necessary to make a difference. We should be more inviting to people that would be happy to spend maybe 1 or 2 hours.

Also, we should spend more time upfront to select which bugs to work on and discuss them on the mailing list so that it is clear what people will work on and maybe also people work on bugs before/after the bug day.

### Agenda

- Follow-up on summit goals
- Review bug day, plan next bug day

### Ongoing issues

Those issues are currently ongoing. We don't have to discuss them. We just need to follow up on them eventually.

- **Meta**
  - Review meeting itself, maybe add extra 15 minutes for "meta" once a month or every two months? (postponed until 2010-06-01)
  - How to organize open issues in the long run (Blueprints? Other tool? Continue text files?)
- Repository policy
- **Test runners / nightly builds**
  - **Supporting Python 2.7**
    - Needs help from the buildbots
  - **Windows**
    - Compiler licenses (Tres, postponed until after 2010-06-14)
  - Build bot organization
- Bug day organization
- **Documentation**
  - **Consolidate "floating" documentation into Sphinx/docs.zope.org**
    - Write blueprint for the consolidation effort (Theuni) see https://blueprints.edge.launchpad.net/zopetoolkit-project/+spec/consolidate-documentation
    - Find candidate links and gather them centrally
    - Edit/update the documentation from the link list and land in Sphinx-style during a sprint
  - **Turn ZTK package documentation into sphinx style (like zope.event)**
    - write bug and assign to toolkit projects (Theuni)
  - Unified index?
- **Releases**
  - How to find a good point when to cut a new release for a package for which fixed bugs where registered (or changes have been made)? Any automation possible to alert us when changes have been sitting around unreleased for a while?

### Topic proposals

- Chris McDonough: Pondering *some* (re-)structuring of the ZTK to allow for better maintenance/release management/communication/marketing.

- Christian Theune: I'd like us to ponder how we can (in addition to the housekeeping and cleanups we do) also move to do constructive work together to expand the stuff that Zope packages (ZTK) is about. How do we go about implementing new technologies together, like supporting HTML 5 in the various parts? I'd like to start putting in new code in the foreseeable future in the zope.* namespace.

### Archive

#### 2010-10-05

This is the agenda and summary for the weekly Zope developer meeting of Tuesday, 2010-10-05 on #zope@irc.freenode.org from 15:00 to 15:30 UTC.

### Summary

**The IRC log is available here:** XXX

### Attendees

XXX

### Agenda

- Follow-up on summit goals
- Review bug day, plan next bug day
- Follow-up on repository policy nagging

### Ongoing issues

Those issues are currently ongoing. We don't have to discuss them. We just need to follow up on them eventually.

- **Meta**
  - Review meeting itself, maybe add extra 15 minutes for "meta" once a month or every two months? (postponed until 2010-06-01)
  - How to organize open issues in the long run (Blueprints? Other tool? Continue text files?)
- **Test runners / nightly builds**
  - **Supporting Python 2.7**
    - Needs help from the buildbots
  - **Windows**
    - Compiler licenses (Tres, postponed until after 2010-06-14)
  - Build bot organization
- Bug day organization
- **Documentation**
  - **Consolidate "floating" documentation into Sphinx/docs.zope.org**

        * Write blueprint for the consolidation effort (Theuni) see https://blueprints.edge.launchpad.net/zopetoolkit-project/+spec/consolidate-documentation
        * Find candidate links and gather them centrally
        * Edit/update the documentation from the link list and land in Sphinx-style during a sprint
    - **Turn ZTK package documentation into sphinx style (like zope.event)**
        * write bug and assign to toolkit projects (Theuni)
    - Unified index?
- **Releases**
    - How to find a good point when to cut a new release for a package for which fixed bugs where registered (or changes have been made)? Any automation possible to alert us when changes have been sitting around unreleased for a while?

### Topic proposals

- Chris McDonough: Pondering *some* (re-)structuring of the ZTK to allow for better maintenance/release management/communication/marketing.
- Christian Theune: I'd like us to ponder how we can (in addition to the housekeeping and cleanups we do) also move to do constructive work together to expand the stuff that Zope packages (ZTK) is about. How do we go about implementing new technologies together, like supporting HTML 5 in the various parts? I'd like to start putting in new code in the foreseeable future in the zope.* namespace.

### 2010-09-28

This is the agenda and summary for the weekly Zope developer meeting of Tuesday, 2010-09-28 on #zope@irc.freenode.org from 15:00 to 15:30 UTC.

### Summary

**The IRC log is available here:** http://zope3.pov.lt/irclogs-zope/%23zope.2010-09-28.log.html#t2010-09-28T18:01:27

### Attendees

Charlie Clark, cwarner, Adam Groszer, Marius Gedminas, Patrick Gerken

### Follow-up on summit topics

Charlie asked for the progress on the post-summit tasks. The summary provided by Christian doesn't reflect all the information given during the summit and haven't been compiled into the summary wiki page (http://wiki.zope.org/ztk/ZopeSummit2010Summary), but only state the headlines. This leaves both detailed explanation of the goals as well as the deadlines missing.

Charlie offered to ask Christian about the missing data.

### Agenda

- Follow-up on summit topics

### Ongoing issues

Those issues are currently ongoing. We don't have to discuss them. We just need to follow up on them eventually.

- **Meta**
  - Review meeting itself, maybe add extra 15 minutes for "meta" once a month or every two months? (postponed until 2010-06-01)
  - How to organize open issues in the long run (Blueprints? Other tool? Continue text files?)
- **Test runners / nightly builds**
  - **Supporting Python 2.7**
    - Needs help from the buildbots
  - **Windows**
    - Compiler licenses (Tres, postponed until after 2010-06-14)
  - Build bot organization
- Bug day organization
- **Documentation**
  - **Consolidate "floating" documentation into Sphinx/docs.zope.org**
    - Write blueprint for the consolidation effort (Theuni) see https://blueprints.edge.launchpad.net/zopetoolkit-project/+spec/consolidate-documentation
    - Find candidate links and gather them centrally
    - Edit/update the documentation from the link list and land in Sphinx-style during a sprint
  - **Turn ZTK package documentation into sphinx style (like zope.event)**
    - write bug and assign to toolkit projects (Theuni)
  - Unified index?
- **Releases**
  - How to find a good point when to cut a new release for a package for which fixed bugs where registered (or changes have been made)? Any automation possible to alert us when changes have been sitting around unreleased for a while?

### Topic proposals

- Chris McDonough: Pondering *some* (re-)structuring of the ZTK to allow for better maintenance/release management/communication/marketing.
- Christian Theune: I'd like us to ponder how we can (in addition to the housekeeping and cleanups we do) also move to do constructive work together to expand the stuff that Zope packages (ZTK) is about. How do we go about implementing new technologies together, like supporting HTML 5 in the various parts? I'd like to start putting in new code in the foreseeable future in the zope.* namespace.

### 2010-09-21

This is the agenda and summary for the weekly Zope developer meeting of Tuesday, 2010-09-21 on #zope@irc.freenode.org from 15:00 to 15:30 UTC.

### Summary

**The IRC log is available here:** http://zope3.pov.lt/irclogs-zope/%23zope.2010-09-21.log.html#t2010-09-21T18:00:27

### Attendees

Charlie Clark, Adam Groszer, Christian Theune, Hanno Schlichting, Fred Drake,

### Post-summit thoughts

Charlie thanked Christian for the organization of the [gocept] party, Zope summit and DZUG conference.

Hanno noted that he shared his thoughts with the rest of Jarn. Publicly they're happy to have not used Zope 3 and grok but find Plone and repoze.bfg being good choices as their "platforms".

Christian reported that he started an experiment together with Wolfgang to take what we have with the ZCA and apply this and other patterns (specifically the "Ports and adapters" pattern) to form a set of procedures and software to help developers to better model their applications and stay flexible with the technical environment their application lives in.

The summit also produced a couple of specific smaller goals that people signed up for dealing with in the near future. They are listed in the wiki page: http://wiki.zope.org/ztk/ZopeSummit2010Summary

### Bug day organization

The next bug day will be 2010-09-29 and was announced on the list by mail during the meeting.

### Meta

Charlie noticed that this meeting was unstructured but we still got good discussions out of it. The defined time-slot seems to help. Going forward we need to deal with the outcome of the summit.

Christian noted that he feels uncomfortable just randomly picking topics. As long as nobody complains the choices are probably fine, though.

As the reminders get dropped every now and then the goal for sending them out is moved to Friday: giving people more time to think about the agenda and Christian more time to procrastinate but still make it ... ;)

### Agenda

- Post-summit thoughts
- Bug day organization
- Meta

### Ongoing issues

Those issues are currently ongoing. We don't have to discuss them. We just need to follow up on them eventually.

- **Meta**
  - Review meeting itself, maybe add extra 15 minutes for "meta" once a month or every two months? (postponed until 2010-06-01)
  - How to organize open issues in the long run (Blueprints? Other tool? Continue text files?)
- **Test runners / nightly builds**
  - **Supporting Python 2.7**
    - Needs help from the buildbots
  - **Windows**
    - Compiler licenses (Tres, postponed until after 2010-06-14)
  - Build bot organization
- Bug day organization
- **Documentation**
  - **Consolidate "floating" documentation into Sphinx/docs.zope.org**
    - Write blueprint for the consolidation effort (Theuni) see https://blueprints.edge.launchpad.net/zopetoolkit-project/+spec/consolidate-documentation
    - Find candidate links and gather them centrally
    - Edit/update the documentation from the link list and land in Sphinx-style during a sprint
  - **Turn ZTK package documentation into sphinx style (like zope.event)**
    - write bug and assign to toolkit projects (Theuni)
  - Unified index?
- **Releases**
  - How to find a good point when to cut a new release for a package for which fixed bugs where registered (or changes have been made)? Any automation possible to alert us when changes have been sitting around unreleased for a while?

### Topic proposals

- Chris McDonough: Pondering *some* (re-)structuring of the ZTK to allow for better maintenance/release management/communication/marketing.
- Christian Theune: I'd like us to ponder how we can (in addition to the housekeeping and cleanups we do) also move to do constructive work together to expand the stuff that Zope packages (ZTK) is about. How do we go about implementing new technologies together, like supporting HTML 5 in the various parts? I'd like to start putting in new code in the foreseeable future in the zope.* namespace.

### 2010-08-31

This is the agenda and summary for the weekly Zope developer meeting of Tuesday, 2010-08-31 on #zope@irc.freenode.org from 15:00 to 15:30 UTC.

### Summary

**The IRC log is available here:** http://zope3.pov.lt/irclogs-zope/%23zope.2010-08-31.log.html#t2010-08-31T18:00:25

### Attendees

Charlie Clark, Adam Groszer, Patrick Gerken

### Agenda

- Supporting Python 2.7
  - Needs help from the buildbots

    Patrick reported that ZODB + Windows is still struggling with Python 2.7. Windows support remains problem for Zope with apparently few core developers using it.

    It was agreed that full support for Python 2.7 is still some way away, probably ZTK 1.1 but this is a matter for the ZTK team.
- Abandoned projects (Tres)

  Charlie suggested that to avoid overly subjective project dropping, criteria for being included need defining "I don't use XYZ so I don't care about it". Such criteria might be the quality and scope of test coverage and technical documentation, an area where Zope traditionally compares unfavourably with other frameworks.

  It was agreed that no projects should be dumped unceremoniously and that a period of grace should be given to allow consumers of projects that have no maintainer to raise their concerns.

  Improvements in the presentation of test results should hopefully make it easier to identify potentiatlly "weak" projects and maybe even inspire developers to improve their tests. Any inclusion criteria should be fairly strictly applied for new projects in the hope of establishing a virtuous spiral. This topic should be continued at the Zope summit in Halle on 13th September.
- Windows
  - Compiler licenses

    Everyone has now received the MS Visual Studio compiler licences.

### Ongoing issues

Those issues are currently ongoing. We don't have to discuss them. We just need to follow up on them eventually.

- Meta
  - Review meeting itself, maybe add extra 15 minutes for "meta" once a month or every two months? (postponed until 2010-06-01)
  - How to organize open issues in the long run (Blueprints? Other tool? Continue text files?)
- Test runners / nightly builds
  - Supporting Python 2.7
    - Needs help from the buildbots
  - Windows

        * Compiler licenses (Tres, postponed until after 2010-06-14)
    - Build bot organization
- Bug day organization
- Documentation
    - Consolidate "floating" documentation into Sphinx/docs.zope.org
        * Write blueprint for the consolidation effort (Theuni). See https://blueprints.edge.launchpad.net/zopetoolkit-project/+spec/consolidate-documentation
        * Find candidate links and gather them centrally
        * Edit/update the documentation from the link list and land in Sphinx-style during a sprint
    - Turn ZTK package documentation into sphinx style (like zope.event)
        * write bug and assign to toolkit projects (Theuni)
    - Unified index?
- Releases
    - How to find a good point when to cut a new release for a package for which fixed bugs where registered (or changes have been made)? Any automation possible to alert us when changes have been sitting around unreleased for a while?

### Topic proposals

- Chris McDonough: Pondering *some* (re-)structuring of the ZTK to allow for better maintenance/release management/communication/marketing.
- Christian Theune: I'd like us to ponder how we can (in addition to the housekeeping and cleanups we do) also move to do constructive work together to expand the stuff that Zope packages (ZTK) is about. How do we go about implementing new technologies together, like supporting HTML 5 in the various parts? I'd like to start putting in new code in the foreseeable future in the zope.* namespace.

### 2010-08-24

This is the agenda and summary for the weekly Zope developer meeting of Tuesday, 2010-08-24 on #zope@irc.freenode.org from 15:00 to 15:30 UTC.

### Summary

**The IRC log is available here:** http://zope3.pov.lt/irclogs-zope/%23zope.2010-08-24.log.html

### Attendees

Hanno Schlichting, Charlie Clark, Christian Theune, Patrick Gerken, Adam Groszer

### Last bug day

Although many people answered the Doodle for the bug day in August the actual participation was quite low. It seems that most people had unexpected urgent customer issues to deal with on that specific day.

Only Michael Howitz and Jens Vagelpohl responded to the request for reports on the mailing list.

As September will feature the DZUG conference and the Zope summit the next bug day will be scheduled a little further down the calendar to avoid clashes with travel plans and people needing to catch up with everyday business after travelling.

### Buildbot organization

Adam asked for talking about organizing the buildbots more so that people can get a better overview of all the builds.

Patrick already experimented with a script that fetches data from the buildbots and creates an overarching display (screenshot: http://picasaweb.google.de/lh/photo/jUOVCcnJEV2KeYI9R_pUCMdjTw5Dqg3R_WEjti4Vahk?feat=twitter)

A small terminology debate occured where Patrick noted that people tend to use both 'dev' and 'trunk' in the buildbots. Hanno brought the argument that 'dev' is what we suffix development releases with when denoting a version and thus 'dev'. In comparison to 'trunk', 'dev' is also agnostic to the VCS and was thus being favored in general.

Christian got the code for the aggregator script that fetches information from mail.zope.org and assembles the daily messages to zope-dev. It has been checked into his sandbox in Subversion. We can now work on an improved version of that script.

### Agenda

- Last bug day
- Organizing buildbots

### Ongoing issues

Those issues are currently ongoing. We don't have to discuss them. We just need to follow up on them eventually.

- Abandoned projects (Tres)
- **Meta**
  - Review meeting itself, maybe add extra 15 minutes for "meta" once a month or every two months? (postponed until 2010-06-01)
  - How to organize open issues in the long run (Blueprints? Other tool? Continue text files?)
- **ZTK status**
  - **Towards a ZTK release**
    - Documentation
    - Release scope
- **Test runners / nightly builds**
  - **Supporting Python 2.7**
    - Needs help from the buildbots

    - **Windows**
        * Compiler licenses (Tres, postponed until after 2010-06-14)
- **Documentation**
    - **Consolidate "floating" documentation into Sphinx/docs.zope.org**
        * Write blueprint for the consolidation effort (Theuni) see https://blueprints.edge.launchpad.net/zopetoolkit-project/+spec/consolidate-documentation
        * Find candidate links and gather them centrally
        * Edit/update the documentation from the link list and land in Sphinx-style during a sprint
    - **Turn ZTK package documentation into sphinx style (like zope.event)**
        * write bug and assign to toolkit projects (Theuni)
    - Unified index?
- **Releases**
    - How to find a good point when to cut a new release for a package for which fixed bugs where registered (or changes have been made)? Any automation possible to alert us when changes have been sitting around unreleased for a while?

### Topic proposals

- Chris McDonough: Pondering *some* (re-)structuring of the ZTK to allow for better maintenance/release management/communication/marketing.
- Christian Theune: I'd like us to ponder how we can (in addition to the housekeeping and cleanups we do) also move to do constructive work together to expand the stuff that Zope packages (ZTK) is about. How do we go about implementing new technologies together, like supporting HTML 5 in the various parts? I'd like to start putting in new code in the foreseeable future in the zope.* namespace.

### 2010-08-17

This is the agenda and summary for the weekly Zope developer meeting of Tuesday, 2010-08-17 on #zope@irc.freenode.org from 15:00 to 15:30 UTC.

### Summary

**The IRC log is available here:** XXX

### Attendees

XXX

### Agenda

- Upcoming bug day
- Reduce noise from test runs (try getting to a green bar)

- How to choose topics

### Ongoing issues

Those issues are currently ongoing. We don't have to discuss them. We just need to follow up on them eventually.

- Abandoned projects (Tres)
- **Meta**
  - Review meeting itself, maybe add extra 15 minutes for "meta" once a month or every two months? (postponed until 2010-06-01)
  - How to organize open issues in the long run (Blueprints? Other tool? Continue text files?)
- **ZTK status**
  - **Towards a ZTK release**
    - Documentation
    - Release scope
- **Test runners / nightly builds**
  - **Supporting Python 2.7**
    - Needs help from the buildbots
  - **Windows**
    - Compiler licenses (Tres, postponed until after 2010-06-14)
- **Documentation**
  - **Consolidate "floating" documentation into Sphinx/docs.zope.org**
    - Write blueprint for the consolidation effort (Theuni) see https://blueprints.edge.launchpad.net/zopetoolkit-project/+spec/consolidate-documentation
    - Find candidate links and gather them centrally
    - Edit/update the documentation from the link list and land in Sphinx-style during a sprint
  - **Turn ZTK package documentation into sphinx style (like zope.event)**
    - write bug and assign to toolkit projects (Theuni)
  - Unified index?
- **Releases**
  - How to find a good point when to cut a new release for a package for which fixed bugs where registered (or changes have been made)? Any automation possible to alert us when changes have been sitting around unreleased for a while?

### Topic proposals

- Chris McDonough: Pondering *some* (re-)structuring of the ZTK to allow for better maintenance/release management/communication/marketing.

- Christian Theune: I'd like us to ponder how we can (in addition to the housekeeping and cleanups we do) also move to do constructive work together to expand the stuff that Zope packages (ZTK) is about. How do we go about implementing new technologies together, like supporting HTML 5 in the various parts? I'd like to start putting in new code in the foreseeable future in the zope.* namespace.

#### 2010-08-10

This is the agenda and summary for the weekly Zope developer meeting of Tuesday, 2010-08-10 on #zope@irc.freenode.org from 15:00 to 15:30 UTC.

#### Summary

**The IRC log is available here:** http://zope3.pov.lt/irclogs-zope/%23zope.2010-08-10.log.html#t2010-08-10T17:57:52

#### Attendees

Charlie Clark, Christian Theune, Adam Groszer, Matthew Wilkes, Patrick Gerken,

#### DZUG conference / summit preparation

Charlie mentioned that the conference could use a bit more publicity. He volunteered to ensure we geta link from pycon.org to the conference homepage. Christian Theune agreed to send mails to the related mailing lists that were mentioned.

We also pointed out the programme to the others present at the meeting and answered some specific questions including that some parts of the programme will be in English but the major part is held in German. We won't make video recordings of the talk as we aren't prepared to deal with neither the technology nor the legal implications.

The summit preparation is getting along: we're currently making sure every participant has a hotel room and is getting his travels and maybe participation with the DZUG conference sorted out. Christian is hoping to get a rough list of topics within the next days.

#### Test runners vs. noise

Christian feels that the test aggregator currently contains too much noise and is thus disregarded at times. A two-fold improvement was suggested: make the mail body contain less information (only the subjects of the failed tests) and provide a single link to an HTML page that displays the results of all tests in a stable sorting. Christian agreed to contact Stefan Holek to find out about the existing code so we can improve it.

Charlie also recommended looking at a talk that was held at EP this year: http://wiki.europython.eu/Talks/Visualizing%20Software%20Quality

#### Wineggbuilder

Adam has published documentation about the windows buildbot/wineggbuilder that is paid for by the foundation: http://docs.zope.org/zopetoolkit/process/winbotsetup.html

In addition to the technical details Adam agreed to provide some introductory information that tells developers what this service is for and how to deal with it.

The egg builder currently runs once a day. As it takes only 40 seconds during an empty run we decided to increase the frequency to every 30 minutes.

### Documentation

Christian noted that he managed to write the bug for cleaning up documentation and the blueprint for gathering floating documentation.

### Agenda

- Update on Zope summit and DZUG conference
- **Test runners / nightly builds**
  - Reduce noise from test runs (try getting to a green bar)
  - **Windows**
    - Compiler licenses (Tres, postponed until after 2010-06-14)
    - Win egg builder (Adam)
    - Documentation about VM setup (Adam)
- **Documentation**
  - **Consolidate "floating" documentation into Sphinx/docs.zope.org**
    - Write blueprint for the consolidation effort (Theuni) see https://blueprints.edge.launchpad.net/zopetoolkit-project/+spec/consolidate-documentation
    - Find candidate links and gather them centrally
    - Edit/update the documentation from the link list and land in Sphinx-style during a sprint
  - **Turn ZTK package documentation into sphinx style (like zope.event)**
    - write bug and assign to toolkit projects (Theuni)
  - Provide package documentation under docs.zope.org/<packagename> and keep updated based on the projects' trunks. (Jens)
  - Unified index?

### Ongoing issues

Those issues are currently ongoing. We don't have to discuss them. We just need to follow up on them eventually.

- Abandoned projects (Tres)
- **Meta**
  - Review meeting itself, maybe add extra 15 minutes for "meta" once a month or every two months? (postponed until 2010-06-01)
  - How to organize open issues in the long run (Blueprints? Other tool? Continue text files?)
- **ZTK status**
  - **Towards a ZTK release**
    - Documentation

        * Release scope

- **Test runners / nightly builds**
    - **Supporting Python 2.7**
        * Needs help from the buildbots
- **Releases**
    - How to find a good point when to cut a new release for a package for which fixed bugs where registered (or changes have been made)? Any automation possible to alert us when changes have been sitting around unreleased for a while?

### Topic proposals

- Chris McDonough: Pondering *some* (re-)structuring of the ZTK to allow for better maintenance/release management/communication/marketing.
- Christian Theune: I'd like us to ponder how we can (in addition to the housekeeping and cleanups we do) also move to do constructive work together to expand the stuff that Zope packages (ZTK) is about. How do we go about implementing new technologies together, like supporting HTML 5 in the various parts? I'd like to start putting in new code in the foreseeable future in the zope.* namespace.

## 2010-08-03

This is the agenda and summary for the weekly Zope developer meeting of Tuesday, 2010-08-03 on #zope@irc.freenode.org from 15:00 to 15:30 UTC.

### Summary

**The IRC log is available here:** http://zope3.pov.lt/irclogs-zope/%23zope.2010-08-03.log.html#t2010-08-03T17:59:16

### Attendees

Stephan Richter, Adam Groszer, Patrick Gerken, Charlie Clark, Christian Theune, davisagli, Marius Gedminas, Baiju Muthudakan

### Documentation

Christian pointed out that Jens has been working on including the packages' documentation on doc.zope.org and that it would be good to pick up the other end (improving existing documentation) now. Christian has little time currently to write the blueprints/bugs to coordinate that effort and is asking for help.

Charlie suggested that working on documentation may be a good sprint topic because people may not want to do it on their own.

Marius asked whether we can have a unified index of our code base ("Were does method *getAdapter* come from?").

### Test runners

Christian asked whether all developers that were supposed to get an MSDN license have received it by now. Adam did but doesn't know about the others. We need to ask around, maybe Tres knows.

Adam considers the Windows egg builder and its documentation to be finished. Due to issues with doc.zope.org and PyPI being down we couldn't have a look at it right now. We'll wait for next week.

### Agenda

- **Documentation**
  - **Consolidate "floating" documentation into Sphinx/docs.zope.org**
    - write blueprint for the consolidation effort (Theuni)
    - Find candidate links and gather them centrally
    - Edit/update the documentation from the link list and land in Sphinx-style during a sprint
  - **Turn ZTK package documentation into sphinx style (like zope.event)**
    - write bug and assign to toolkit projects (Theuni)
  - Provide package documentation under docs.zope.org/<packagename> and keep updated based on the projects' trunks. (Jens)
  - Unified index?
- **Test runners / nightly builds**
  - **Windows**
    - Compiler licenses (Tres, postponed until after 2010-06-14)
    - Win egg builder (Adam)
    - Documentation about VM setup (Adam)

### Ongoing issues

Those issues are currently ongoing. We don't have to discuss them. We just need to follow up on them eventually.

- Abandoned projects (Tres)
- Expectations for the upcoming Zope summit?
- **Meta**
  - Review meeting itself, maybe add extra 15 minutes for "meta" once a month or every two months? (postponed until 2010-06-01)
  - How to organize open issues in the long run (Blueprints? Other tool? Continue text files?)
- **ZTK status**
  - **Towards a ZTK release**
    - Documentation
    - Release scope
- **Test runners / nightly builds**

    - **Supporting Python 2.7**
        * Needs help from the buildbots

- **Releases**
    - How to find a good point when to cut a new release for a package for which fixed bugs where registered (or changes have been made)? Any automation possible to alert us when changes have been sitting around unreleased for a while?

### Topic proposals

- Chris McDonough: Pondering *some* (re-)structuring of the ZTK to allow for better maintenance/release management/communication/marketing.
- Christian Theune: I'd like us to ponder how we can (in addition to the housekeeping and cleanups we do) also move to do constructive work together to expand the stuff that Zope packages (ZTK) is about. How do we go about implementing new technologies together, like supporting HTML 5 in the various parts? I'd like to start putting in new code in the foreseeable future in the zope.* namespace.

### 2010-07-27

This is the agenda and summary for the weekly Zope developer meeting of Tuesday, 2010-07-27 on #zope@irc.freenode.org from 15:00 to 15:30 UTC.

### Summary

**The IRC log is available here:** http://zope3.pov.lt/irclogs-zope/%23zope.2010-07-27.log.html#t2010-07-27T17:59:11

### Attendees

Leonardo Rochael, Wichert Akkerman, Jens Vagelpohl, Adam Groszer, Christian Theune

### Review EuroPython

Unfortunately no EP participant was around and I'd like to encourage those who visited EP to give an update to the non-attending Zope developers on what happened from a Zope perspective. Talks? Sprints? Interesting discussions?

Christian agreed to send a mail inviting participants to write up their experience.

### Review languishing bugs

I retried the experiment inviting the other participants for reviewing languishing bugs. Instead of the misunderstanding of last week a more in-depth discussion about expectations WRT communication via bug trackers, responsiveness, process occured. The discussion was undirected and relatively general but raised some concerns that might be good input for the upcoming summit, too. Please see the IRC log if you're interested in the details.

### Agenda

- Review EuroPython
- Review languishing bugs. (See https://mail.zope.org/pipermail/zope-tests/2010-July/016049.html https://mail.zope.org/pipermail/zope-tests/2010-July/016050.html https://mail.zope.org/pipermail/zope-tests/2010-July/016051.html https://mail.zope.org/pipermail/zope-tests/2010-July/016052.html)

### Ongoing issues

Those issues are currently ongoing. We don't have to discuss them. We just need to follow up on them eventually.

- Abandoned projects (Tres)
- Expectations for the upcoming Zope summit?
- **Meta**
  - Review meeting itself, maybe add extra 15 minutes for "meta" once a month or every two months? (postponed until 2010-06-01)
  - How to organize open issues in the long run (Blueprints? Other tool? Continue text files?)
- **ZTK status**
  - **Towards a ZTK release**
    - Documentation
    - Release scope
- **Test runners / nightly builds**
  - **Windows**
    - Compiler licenses (Tres, postponed until after 2010-06-14)
    - Win egg builder (Adam)
    - Documentation about VM setup (Adam)
  - **Supporting Python 2.7**
    - Needs help from the buildbots
- **Documentation**
  - **Consolidate "floating" documentation into Sphinx/docs.zope.org**
    - write blueprint for the consolidation effort (Theuni)
    - Find candidate links and gather them centrally
    - Edit/update the documentation from the link list and land in Sphinx-style during a sprint
  - **Turn ZTK package documentation into sphinx style (like zope.event)**
    - write bug and assign to toolkit projects (Theuni)
  - Provide package documentation under docs.zope.org/<packagename> and keep updated based on the projects' trunks. (Jens)
- **Releases**

  - How to find a good point when to cut a new release for a package for which fixed bugs where registered (or changes have been made)? Any automation possible to alert us when changes have been sitting around unreleased for a while?

### Topic proposals

- Chris McDonough: Pondering *some* (re-)structuring of the ZTK to allow for better maintenance/release management/communication/marketing.
- Christian Theune: I'd like us to ponder how we can (in addition to the housekeeping and cleanups we do) also move to do constructive work together to expand the stuff that Zope packages (ZTK) is about. How do we go about implementing new technologies together, like supporting HTML 5 in the various parts? I'd like to start putting in new code in the foreseeable future in the zope.* namespace.

### 2010-07-20

This is the agenda and summary for the weekly Zope developer meeting of Tuesday, 2010-07-20 on #zope@irc.freenode.org from 15:00 to 15:30 UTC.

### Summary

**The IRC log is available here:** http://zope3.pov.lt/irclogs-zope/%23zope.2010-07-20.log.html#t2010-07-20T17:58:02

### Attendees

Jens Vagelpohl, Adam Groszer, Christian Theune, Leo Rochael

### Bugday review and planning

Jens Vagelpohl worked on fixing Zope 2 bugs (fixed 5 and worked on 4 others) and Tres Seaver tried fixing an issue with zope.testing under Python 2.7.

The last bug day has been relatively silent because only few participants signed up some of them didn't make it. It appears that both holiday season and scheduling issues kept people from joining.

Jens noted that the wiki page is cumbersome to deal with due to the ReST table layout. Christian agreed to adapt to a regular list for the next wiki page.

Christian will doodle the next bug day for the week of August 16-20.

### Languishing bug review

Unfortunately this part of the agenda didn't work out as expected. I tried to take some time so that everyone present would review some bugs, not talking about reviewing bugs. I miscommunicated here and talked about some things while I reviewed bugs thinking everyone did the same ...

Jens reported that Hanno has been and is triaging the Zope 2 bugs and he is working on the things that can be fixed, too.

We also noted that the term "languish" is not understood the same: the current meaning in the way the test scripts use it is "new bugs that nobody triaged". The goal for avoiding languishing bugs is to give feedback to bug reporters in a timely manner. That doesn't necessarily mean we fix them right away but we can respond whether it is indeed a bug, or misunderstanding or a feature request. This should give our users a better feeling on how we are going to deal with an issue in the long run.

I asked for a reminder of whether we agreed to put minimum version dependency requirements into setup.py or not. Nobody could remember and we couldn't produce reference material on a decision but we think that minimum requirements were agreed upon to be ok.

### Agenda

- Review last bug day, plan for next
- Review languishing bugs. (See https://mail.zope.org/pipermail/zope-tests/2010-July/016049.html, https://mail.zope.org/pipermail/zope-tests/2010-July/016050.html, https://mail.zope.org/pipermail/zope-tests/2010-July/016051.html, https://mail.zope.org/pipermail/zope-tests/2010-July/016052.html)

### Ongoing issues

Those issues are currently ongoing. We don't have to discuss them. We just need to follow up on them eventually.

- Abandoned projects (Tres)
- Expectations for the upcoming Zope summit?
- **Meta**
  - Review meeting itself, maybe add extra 15 minutes for "meta" once a month or every two months? (postponed until 2010-06-01)
  - How to organize open issues in the long run (Blueprints? Other tool? Continue text files?)
- **ZTK status**
  - **Towards a ZTK release**
    - Documentation
    - Release scope
- **Test runners / nightly builds**
  - **Windows**
    - Compiler licenses (Tres, postponed until after 2010-06-14)
    - Win egg builder (Adam)
    - Documentation about VM setup (Adam)
  - **Supporting Python 2.7**
    - Needs help from the buildbots
- **Documentation**
  - **Consolidate "floating" documentation into Sphinx/docs.zope.org**
    - write blueprint for the consolidation effort (Theuni)
    - Find candidate links and gather them centrally

        * Edit/update the documentation from the link list and land in Sphinx-style during a sprint
    - **Turn ZTK package documentation into sphinx style (like zope.event)**
        * write bug and assign to toolkit projects (Theuni)
    - Provide package documentation under docs.zope.org/<packagename> and keep updated based on the projects' trunks. (Jens)
- **Releases**
    - How to find a good point when to cut a new release for a package for which fixed bugs where registered (or changes have been made)? Any automation possible to alert us when changes have been sitting around unreleased for a while?

### Topic proposals

- Chris McDonough: Pondering *some* (re-)structuring of the ZTK to allow for better maintenance/release management/communication/marketing.
- Christian Theune: I'd like us to ponder how we can (in addition to the housekeeping and cleanups we do) also move to do constructive work together to expand the stuff that Zope packages (ZTK) is about. How do we go about implementing new technologies together, like supporting HTML 5 in the various parts? I'd like to start putting in new code in the foreseeable future in the zope.* namespace.

### 2010-07-13

This is the agenda and summary for the weekly Zope developer meeting of Tuesday, 2010-07-13 on #zope@irc.freenode.org from 15:00 to 15:30 UTC.

### Summary

**The IRC log is available here:** http://zope3.pov.lt/irclogs-zope/%23zope.2010-07-13.log.html#t2010-07-13T17:57:24

### Attendees

Christian Theune, Charlie Clark, Tres Seaver, Fred Drake, Hanno Schlichting

### Abandoned projects

Tres reported that there are a bunch of projects in SVN that nobody seems to care about (at least enough to update copyright heads, conform to policy, ...). In the long run those may need to be moved away from the main tree or repository to signal that they're not actively maintained.

We'd like to take action to figure out which of those projects really aren't maintained anymore.

A first step for this would be to make it more visible who the developers are that (used to) maintain the code. The idea came up to enhance the repository policy report with the information of "when was the last commit", "who were the frequent committers for this branch". This would allow developers to quickly scan the list of projects for their name and thus check faster whether any of the projects they are interested in needs attention.

Charlie Clark volunteered to make the necessary changes to the repository policy checking code as soon as he has time in the next week(s).

### ZTK vs Python 2.7

The mailing list thread about ZTK/Python 2.7 was picked up. Tres expressed interest in supporting Python 2.7 ASAP.

For ZTK 1.0 there are multiple reasons not to support Python 2.7:

- ZTK 1.0 still needs to support Python 2.4 and supporting many Python versions in parallel (2.4-2.7) is maybe not a good idea
- ZODB 3.9 isn't compatible and upgrading to 3.10 won't happen before ZTK 1.1

ZTK 1.1 then could drop support for Python 2.4 and upgrade packages to get them in a version that supports Python 2.7.

### Windows help needed

Tres noted that we're notoriously short on Windows developers that help fixing bug. If you're a Windows developer then we would appreciate if you could help fixing bugs quickly that appear on Windows as none of the core developers run Windows as their primary working environment.

### Agenda

- Abandoned projects (Tres)

### Ongoing issues

Those issues are currently ongoing. We don't have to discuss them. We just need to follow up on them eventually.

- Use of $Id$ properties in code (keep, remove active, remove opportunistically) See https://mail.zope.org/pipermail/zope-cmf/2010-July/029208.html
- Expectations for the upcoming Zope summit?
- **Meta**
  - Review meeting itself, maybe add extra 15 minutes for "meta" once a month or every two months? (postponed until 2010-06-01)
  - How to organize open issues in the long run (Blueprints? Other tool? Continue text files?)
- **ZTK status**
  - **Towards a ZTK release**
    - Documentation
    - Release scope
- **Test runners / nightly builds**
  - **Windows**
    - Compiler licenses (Tres, postponed until after 2010-06-14)
    - Win egg builder (Adam)
    - Documentation about VM setup (Adam)
  - **Supporting Python 2.7**
    - Needs help from the buildbots

- **Documentation**
  - **Consolidate "floating" documentation into Sphinx/docs.zope.org**
    - write blueprint for the consolidation effort (Theuni)
    - Find candidate links and gather them centrally
    - Edit/update the documentation from the link list and land in Sphinx-style during a sprint
  - **Turn ZTK package documentation into sphinx style (like zope.event)**
    - write bug and assign to toolkit projects (Theuni)
  - Provide package documentation under docs.zope.org/<packagename> and keep updated based on the projects' trunks. (Jens)
- **Releases**
  - How to find a good point when to cut a new release for a package for which fixed bugs where registered (or changes have been made)? Any automation possible to alert us when changes have been sitting around unreleased for a while?

### Topic proposals

- Chris McDonough: Pondering *some* (re-)structuring of the ZTK to allow for better maintenance/release management/communication/marketing.
- Christian Theune: I'd like us to ponder how we can (in addition to the housekeeping and cleanups we do) also move to do constructive work together to expand the stuff that Zope packages (ZTK) is about. How do we go about implementing new technologies together, like supporting HTML 5 in the various parts? I'd like to start putting in new code in the foreseeable future in the zope.* namespace.

### 2010-07-06

This is the agenda and summary for the weekly Zope developer meeting of Tuesday, 2010-07-06 on #zope@irc.freenode.org from 15:00 to 15:30 UTC.

### Summary

**The IRC log is available here:** http://zope3.pov.lt/irclogs-zope/%23zope.2010-07-06.log.html#t2010-07-06T18:00:42

### Attendees

Tres Seaver, Adam Groszer, Hanno Schlichting, Charlie Clark, Wichert Akkermann, Jens Vagelpohl, d2m, davisagli

### VCS properties

In a recent discussion on the CMF mailinglist the question was raised what our ruleso n using VCS properties (especially $Id$) are.

The discussion traced the origin back to CVS and generally argued that the need for the IDs does not exist anymore but actually obstruct diffs in SVN. The suggestion was to change the policy to ask for not adding properties in the future anymore and opportunistically remove IDs whenever we stumble over them.

A poll showed 7 people in favor of this action, none against it. Tres updated the ZTK documentation to reflect the decision.

### Expectations for the summit

We had a round of people expressing what interest and expectations they have on topics that we might deal with at the summit. Here's the assorted list of topics that came up:

- more compelling stories for Zope
- better documentation
- developer portraits/company profiles/product use-cases on zope.org
- opening up for experimentation and then folding together common experiences into common code (like HTML 5)
- higher quality code versus more code
- actively find and describe use cases

### Optional C-optimisations

Some projects (like i18nmessageid) have optional C-code and in general would work on Non-CPython-platforms (GAE, Jython, IronPython, Windows) if the build process would support it. Unfortunately neither setuptools nor distributes helps with spelling optional extensions, thus Tres started working on zope.optionalextension which is intended to work together with bare distutils, setuptools and distribute.

### Meta

Today we had a surprising large number of participants, compared to previous weeks with only the same three people.

A member from US West Coast reported that he has a hard time joining at the current time the meeting takes place at.

Hanno pointed out that we might want to take some topics explicitly to the mailing list if attendance is low and topics need to be driven forward. Charlie pointed out that to get to decisions real-time meetings (like IRC) may be more productive. There was no conclusion on the issue and Hanno withdrew his suggestion.

Christian Theune asked for input about managing ongoing issues, but no ideas came up.

### Agenda

- Use of $Id$ properties in code (keep, remove active, remove opportunistically) See https://mail.zope.org/pipermail/zope-cmf/2010-July/029208.html
- Expectations for the upcoming Zope summit?
- **Meta**
  - Review meeting itself, maybe add extra 15 minutes for "meta" once a month or every two months? (postponed until 2010-06-01)
  - How to organize open issues in the long run (Blueprints? Other tool? Continue text files?)

### Ongoing issues

Those issues are currently ongoing. We don't have to discuss them. We just need to follow up on them eventually.

- **ZTK status**
  - **Towards a ZTK release**
    - Documentation
    - Release scope
- **Test runners / nightly builds**
  - **Windows**
    - Compiler licenses (Tres, postponed until after 2010-06-14)
    - Win egg builder (Adam)
    - Documentation about VM setup (Adam)
  - **Supporting Python 2.7**
    - Needs help from the buildbots
- **Documentation**
  - **Consolidate "floating" documentation into Sphinx/docs.zope.org**
    - write blueprint for the consolidation effort (Theuni)
    - Find candidate links and gather them centrally
    - Edit/update the documentation from the link list and land in Sphinx-style during a sprint
  - **Turn ZTK package documentation into sphinx style (like zope.event)**
    - write bug and assign to toolkit projects (Theuni)
  - Provide package documentation under docs.zope.org/<packagename> and keep updated based on the projects' trunks. (Jens)
- **Releases**
  - How to find a good point when to cut a new release for a package for which fixed bugs where registered (or changes have been made)? Any automation possible to alert us when changes have been sitting around unreleased for a while?

### Topic proposals

- Chris McDonough: Pondering *some* (re-)structuring of the ZTK to allow for better maintenance/release management/communication/marketing.
- Christian Theune: I'd like us to ponder how we can (in addition to the housekeeping and cleanups we do) also move to do constructive work together to expand the stuff that Zope packages (ZTK) is about. How do we go about implementing new technologies together, like supporting HTML 5 in the various parts? I'd like to start putting in new code in the foreseeable future in the zope.* namespace.

### 2010-06-29

This is the agenda and summary for the weekly Zope developer meeting of Tuesday, 2010-06-29 on #zope@irc.freenode.org from 15:00 to 15:30 UTC.

### Summary

**The IRC log is available here:** http://zope3.pov.lt/irclogs-zope/%23zope.2010-06-29.log.html#t2010-06-29T18:00:17

### Attendees

Charlie Clark, Adam Groszer, Jens Vagelpohl, Chris McDonough

### KGS 3.4.1 release

The KGS 3.4.1 was released and Adam was thanked for his work. He pointed out that the checklist for announcements contained a reference to a German magazine that requires a German translation but hasn't found any German willing to do the translation yet. He was pointed to ask on the mailing list.

The topic of the end of life for Zope 3 releases crept up again and it was pointed out that Zope 3.4 is the last major release: there will be no Zope 3.5 but BlueBream.

### Zope summit

Some people threw in some thoughts they had about discussing future Zope developments. Here's a paraphrased list of what was said on IRC:

- Charlie Clark would like to ponder certification on software quality (touching test coverage)
- Charlie Clark would like to see more explanation on how to use a component-based approach for solving problems instead of adding more features
- Adam would like to look around at "others" to find out where Zope is missing out
- Charlie and Chris think that the "bicycle toolkit" should be discussed

### Agenda

- Wrap up/review of KGS 3.4.1 release, any next steps?
- Expectations for the upcoming Zope summit?

### Ongoing issues

Those issues are currently ongoing. We don't have to discuss them. We just need to follow up on them eventually.

- **ZTK status**
  - **Towards a ZTK release**
    - Documentation
    - Release scope
- **Test runners / nightly builds**
  - **Windows**
    - Compiler licenses (Tres, postponed until after 2010-06-14)

        * Win egg builder (Adam)
        * Documentation about VM setup (Adam)
    - **Supporting Python 2.7**
        * Needs help from the buildbots
- **Documentation**
    - **Consolidate "floating" documentation into Sphinx/docs.zope.org**
        * write blueprint for the consolidation effort (Theuni)
        * Find candidate links and gather them centrally
        * Edit/update the documentation from the link list and land in Sphinx-style during a sprint
    - **Turn ZTK package documentation into sphinx style (like zope.event)**
        * write bug and assign to toolkit projects (Theuni)
    - Provide package documentation under docs.zope.org/<packagename> and keep updated based on the projects' trunks. (Jens)
- **Releases**
    - How to find a good point when to cut a new release for a package for which fixed bugs where registered (or changes have been made)? Any automation possible to alert us when changes have been sitting around unreleased for a while?
- **Meta**
    - Review meeting itself, maybe add extra 15 minutes for "meta" once a month or every two months? (postponed until 2010-06-01)
    - How to organize open issues in the long run (Blueprints? Other tool? Continue text files?)

### Topic proposals

- Chris McDonough: Pondering *some* (re-)structuring of the ZTK to allow for better maintenance/release management/communication/marketing.
- Christian Theune: I'd like us to ponder how we can (in addition to the housekeeping and cleanups we do) also move to do constructive work together to expand the stuff that Zope packages (ZTK) is about. How do we go about implementing new technologies together, like supporting HTML 5 in the various parts? I'd like to start putting in new code in the foreseeable future in the zope.* namespace.

### 2010-06-22

This is the agenda and summary for the weekly Zope developer meeting of Tuesday, 2010-06-22 on #zope@irc.freenode.org from 15:00 to 15:30 UTC.

### Summary

**The IRC log is available here:** http://zope3.pov.lt/irclogs-zope/%23zope.2010-06-22.log.html#t2010-06-22T18:01:40

### Attendees

Christian Theune, Jens Vagelpohl, Tres Seaver, Charlie Clark, Marius Gedminas

### Bug tracking

The bug tracker monitoring script which is intended to alert developers about newly reported bugs that are missing attention is in place and also now includes the name of the project or project group that is being checked.

We decided to run those tests only once per week to keep the noise down (we probably won't make too much progress in the near future). We also reduced threshold for bugs being marked as languishing from 14 days to 7 days to cater for the higher latency.

### Documentation

Tres reported that the he landed his proposed documentation changes in the zope.event trunk. He would also like to see all the other packages from the toolkit to have their documentation changed this way and reachable from the toolkit documentation on docs.zope.org. We also need to find a way to update the package documentation regularly based on the packages' trunks.

For the floating documentation we need people to find links to candidate documentation that should be integrated and then have a group of people edit that documentation by evaluating it and update it while turning it over into the Sphinx-based documentation. The second part appears to be suitable for a sprint.

### Bug day

Tres updated the bug day's wiki page gathering the feedback from all developers. Charlie noted that he liked the feedback mails for the better visibility to non-bug-dayers.

Christian will schedule the next bug day by providing a Doodle for the next time including a link to the reports of the last bug day. The next bug day will be in the week of July 12th-18th.

### Agenda

- **Bug tracking**
  - Monitoring tracker status (Charlie Clark, ctheune)
- **Documentation**
  - Consolidate "floating" documentation into Sphinx/docs.zope.org
- **Review bug day**
  - Comments about the bug day
  - Scheduling the next bug day

### Ongoing issues

Those issues are currently ongoing. We don't have to discuss them. We just need to follow up on them eventually.

- **ZTK status**

    - **Towards a ZTK release**
        * Documentation
        * Release scope
- **Test runners / nightly builds**
    - **Windows**
        * Compiler licenses (Tres, postponed until after 2010-06-14)
        * Win egg builder (Adam)
        * Documentation about VM setup (Adam)
    - **Supporting Python 2.7**
        * Needs help from the buildbots
- **Documentation**
    - **Consolidate "floating" documentation into Sphinx/docs.zope.org**
        * write blueprint for the consolidation effort (Theuni)
        * Find candidate links and gather them centrally
        * Edit/update the documentation from the link list and land in Sphinx-style during a sprint
    - **Turn ZTK package documentation into sphinx style (like zope.event)**
        * write bug and assign to toolkit projects (Theuni)
    - Provide package documentation under docs.zope.org/<packagename> and keep updated based on the projects' trunks. (Jens)
- **Releases**
    - How to find a good point when to cut a new release for a package for which fixed bugs where registered (or changes have been made)? Any automation possible to alert us when changes have been sitting around unreleased for a while?
- **Meta**
    - Review meeting itself, maybe add extra 15 minutes for "meta" once a month or every two months? (postponed until 2010-06-01)
    - How to organize open issues in the long run (Blueprints? Other tool? Continue text files?)

### Topic proposals

- Chris McDonough: Pondering *some* (re-)structuring of the ZTK to allow for better maintenance/release management/communication/marketing.
- Christian Theune: I'd like us to ponder how we can (in addition to the housekeeping and cleanups we do) also move to do constructive work together to expand the stuff that Zope packages (ZTK) is about. How do we go about implementing new technologies together, like supporting HTML 5 in the various parts? I'd like to start putting in new code in the foreseeable future in the zope.* namespace.

### 2010-06-15

This is the agenda and summary for the weekly Zope developer meeting of Tuesday, 2010-06-15 on #zope@irc.freenode.org from 15:00 to 15:30 UTC.

### Summary

**The IRC log is available here:** http://zope3.pov.lt/irclogs-zope/%23zope.2010-06-15.log.html#t2010-06-15T18:00:35

### Attendees

Charlie Clark, Adam Groszer, Christian Theune, Jens Vagelpohl

### Test runners / nightly builds

Adam received his MSDN license and was able to retrieve software. He pointed out that the license is valid for a year. After that we need to consider renewal (unless the VS express editions work out).

A survey on zope-dev showed that the other developers who asked for a license didn't receive theirs, yet.

Adam continues to work on the VM setup at rackspace, preparing the eggbuilder and buildout slaves.

### Supporting Python 2.7

Supporting Python 2.7 involves at least the following actions:

- get buildbot owners to run with Python 2.7 additionally
- review RestrictedPython

Charlie Clark points out that he has been running installations using Python 2.7 with no practical problems.

### Metrics for bug days

2010-06-15 features a bug day and we'd like to get a better grasp on what we get out of the bug days. After some discussion on quantitative metrics the decision turned towards a qualitative assessment: Christian Theune will write another invitation to the zope-dev list asking every developer who participated in the bug day to write a short summary of what he did/achieved during the bug day.

The sum of those summaries should provide a good overview to others of what happened.

### Index view for download.zope.org

Jens turned download.zope.org into a standard auto-generated index page which Adam acknowledged as a solution to his (proxied) request.

### Agenda (draft)

- **Test runners / nightly builds**
  - **Windows**
    - Compiler licenses (Tres, postponed until after 2010-06-14)
    - Win egg builder (Adam)
    - Documentation about VM setup (Adam)

    - **Supporting Python 2.7**
        * Needs help from the buildbots
- **Metrics for bug days**
    - Find a way to demonstrate what/how much work happened on a bug day.

### Ongoing issues

Those issues are currently ongoing. We don't have to discuss them. We just need to follow up on them eventually.

- **ZTK status**
    - **Towards a ZTK release**
        * Documentation
        * Release scope
- **KGS 3.4.1 release**
    - Index view for download.zope.org
- **Bug tracking**
    - Monitoring tracker status (Charlie Clark, ctheune)
- **Documentation**
    - Consolidate "floating" documentation into Sphinx/docs.zope.org
- **Releases**
    - How to find a good point when to cut a new release for a package for which fixed bugs where registered (or changes have been made)? Any automation possible to alert us when changes have been sitting around unreleased for a while?
- **Meta**
    - Review meeting itself, maybe add extra 15 minutes for "meta" once a month or every two months? (postponed until 2010-06-01)
    - How to organize open issues in the long run (Blueprints? Other tool? Continue text files?)

### Topic proposals

- Chris McDonough: Pondering *some* (re-)structuring of the ZTK to allow for better maintenance/release management/communication/marketing.
- Christian Theune: I'd like us to ponder how we can (in addition to the housekeeping and cleanups we do) also move to do constructive work together to expand the stuff that Zope packages (ZTK) is about. How do we go about implementing new technologies together, like supporting HTML 5 in the various parts? I'd like to start putting in new code in the foreseeable future in the zope.* namespace.

### 2010-06-08

This is the agenda and summary for the weekly Zope developer meeting of Tuesday, 2010-06-08 on #zope@irc.freenode.org from 15:00 to 15:30 UTC.

### Summary

**The IRC log is available here:** http://zope3.pov.lt/irclogs-zope/%23zope.2010-06-08.log.html#t2010-06-08T18:02:12

### Attendees

Adam Groszer, Charlie Clark, Christian Theune

### KGS 3.4.1

The KGS according to Adam only requires a version bump from release candidate to final. Along the way he updated the release process documentation for the KGS which can be found here: http://wiki.zope.org/zope3/MakingARelease.

Adam will poke Jens about getting a better index view for download.zope.org.

### Test-runners/nightly builds

The funding for a VM at Rackspace was granted by the ZF for a testing period of two months. Adam reported that MS announced the MSDN licenses to be available around 2010-06-14.

The activation of the VM is on hold as long as Adam is working on the software that automatically creates binary eggs for releases (wineggbuilder). He will also write documentation on how the VM was set up.

### Meta

Charlie Clark is now officially the second person organizing the weekly meetings. Christian and Charlie will coordinate directly to keep it going.

A quick question for how people feel about the meeting showed general approvement while also pointing out declining participation. One reason seems to be that we've been treading maintenance issues in the last weeks (KGS 3.4.1, test runners, bug tracking, ...) which may not be a very attractive topic.

Christian expressed he is somewhat uncomfortable opening up more issues in parallel as long as we don't find a better way of managing them. Christian will ask the Launchpad folks whether they know about a way to using LP for managing our agenda.

Additionally we will switch to preparing the agenda on mondays and making them more of an invitation style so that we have a higher chance of people actually stating their topic wishes.

### Agenda

- **KGS 3.4.1 release**
  - Document release procedure (Adam)
  - Index view for download.zope.org
- **Test runners / nightly builds**
  - **Windows machines**
    - Compiler licenses (Tres, postponed until after 2010-06-02)

        * AMI/ (Sidnei, Adam)
        * Amazon funding (Adam, Christian Theune)

- **Meta**
    - Review meeting itself, maybe add extra 15 minutes for "meta" once a month or every two months? (postponed until 2010-06-01)
    - How to organize open issues in the long run (Blueprints? Other tool? Continue text files?)
    - Find second person to run the weekly meetings

### Ongoing issues

Those issues are currently ongoing. We don't have to discuss them. We just need to follow up on them eventually.

- **ZTK status**
    - **Towards a ZTK release**
        * Documentation
        * Release scope
- **Bug tracking**
    - Monitoring tracker status (Charlie Clark, ctheune)
- **Documentation**
    - Consolidate "floating" documentation into Sphinx/docs.zope.org
- **Releases**
    - How to find a good point when to cut a new release for a package for which fixed bugs where registered (or changes have been made)? Any automation possible to alert us when changes have been sitting around unreleased for a while?
- **Metrics for bug days**
    - Find a way to demonstrate what/how much work happened on a bug day.

### Topic proposals

- Chris McDonough: Pondering *some* (re-)structuring of the ZTK to allow for better maintenance/release management/communication/marketing.
- Christian Theune: I'd like us to ponder how we can (in addition to the housekeeping and cleanups we do) also move to do constructive work together to expand the stuff that Zope packages (ZTK) is about. How do we go about implementing new technologies together, like supporting HTML 5 in the various parts? I'd like to start putting in new code in the foreseeable future in the zope.* namespace.

### 2010-06-01

This is the agenda and summary for the weekly Zope developer meeting of Tuesday, 2010-06-01 on #zope@irc.freenode.org from 15:00 to 15:30 UTC.

### Agenda

- **Documentation**
  - Consolidate "floating" documentation into Sphinx/docs.zope.org
- **Releases**
  - How to find a good point when to cut a new release for a package for which fixed bugs where registered (or changes have been made)? Any automation possible to alert us when changes have been sitting around unreleased for a while?

### Ongoing issues

Those issues are currently ongoing. We don't have to discuss them. We just need to follow up on them eventually.

- **ZTK status**
  - **Towards a ZTK release**
    - Documentation
    - Release scope
- **KGS 3.4.1 release**
  - Document release procedure (Adam)
  - Index view for download.zope.org
- **Test runners / nightly builds**
  - **Windows machines**
    - Compiler licenses (Tres, postponed until after 2010-06-02)
    - AMI/ (Sidnei, Adam)
    - Amazon funding (Adam, Christian Theune)
- **Bug tracking**
  - Monitoring tracker status (Charlie Clark, ctheune)
- **Documentation**
  - Consolidate "floating" documentation into Sphinx/docs.zope.org
- **Releases**
  - How to find a good point when to cut a new release for a package for which fixed bugs where registered (or changes have been made)? Any automation possible to alert us when changes have been sitting around unreleased for a while?
- **Metrics for bug days**
  - Find a way to demonstrate what/how much work happened on a bug day.
- **Meta**
  - Review meeting itself, maybe add extra 15 minutes for "meta" once a month or every two months? (postponed until 2010-06-01)
  - How to organize open issues in the long run (Blueprints? Other tool? Continue text files?)
  - Find second person to run the weekly meetings

### Topic proposals

- Chris McDonough: Pondering *some* (re-)structuring of the ZTK to allow for better maintenance/release management/communication/marketing.
- Christian Theune: I'd like us to ponder how we can (in addition to the housekeeping and cleanups we do) also move to do constructive work together to expand the stuff that Zope packages (ZTK) is about. How do we go about implementing new technologies together, like supporting HTML 5 in the various parts? I'd like to start putting in new code in the foreseeable future in the zope.* namespace.

### Summary

**The IRC log is available here:** http://zope3.pov.lt/irclogs-zope/%23zope.2010-06-01.log.html#t2010-06-01T18:05:03

### Attendees

Charlie Clark, Hanno Schlichting, Adam Groszer, Tres Seaver

### Documentation

Consolidate "floating" documentation into Sphinx/docs.zope.org

Lots of useful Zope documentation is spread around so that it is difficult to find. Tres suggested that we start with the wikis and that much of the work is editorial - sifting, correcting, updating and deleting.

http://sphinx.pocoo.org/ is a good starting place for learning about Sphinx

### Releases

How to find a good point when to cut a new release for a package for which fixed bugs where registered (or changes have been made)? Any automation possible to alert us when changes have been sitting around unreleased for a while?

The current release process is documented at http://docs.zope.org/zopetoolkit/process/releasing-software.html zest.releaser and jarn.mkrelease are two tools that can be used to make releasing easier.

The zope.bugtracker script can be used to detect likely packages, eg. $ bin/check-bugs -s "Fix Committed" -g zopetoolkit

There was no agreement to get a cronjob to pester developers to make releases for packages where fixes have been committed but it would seem a useful practice after bug days.

### 2010-05-25

This is the agenda and summary for the weekly Zope developer meeting of Tuesday, 2010-05-25 on #zope@irc.freenode.org from 15:00 to 15:30 UTC.

### Agenda

- **Windows builds**
  - AMI (Sidnei, Adam)
  - Amazon funding (Adam, Christian Theune)
- **KGS 3.4.1 release**
  - Document release procedure (Adam)
  - Index view for download.zope.org
- Review Bugday, plan next

### Ongoing issues

Those issues are currently ongoing. We don't have to discuss them. We just need to follow up on them eventually.

- **ZTK status**
  - **Towards a ZTK release**
    - Documentation
    - Release scope
- **Test runners / nightly builds**
  - **Windows machines**
    - Compiler licenses (Tres, postponed until after 2010-06-02)
- **Bug tracking**
  - Monitoring tracker status (Charlie Clark, ctheune)
- **Documentation**
  - Consolidate "floating" documentation into Sphinx/docs.zope.org
- **Releases**
  - How to find a good point when to cut a new release for a package for which fixed bugs where registered (or changes have been made)? Any automation possible to alert us when changes have been sitting around unreleased for a while?
- **Metrics for bug days**
  - Find a way to demonstrate what/how much work happened on a bug day.
- **Meta**
  - Review meeting itself, maybe add extra 15 minutes for "meta" once a month or every two months? (postponed until 2010-06-01)
  - How to organize open issues in the long run (Blueprints? Other tool? Continue text files?)
  - Find second person to run the weekly meetings

### Topic proposals

- Lennart: Of course what applies to Hanno should apply to others making releases of packages maintained by the Zope Toolkit project as well. I think the ZTK leadership should figure out some kind of guidelines for this that people can follow.
- Chris McDonough: Pondering *some* (re-)structuring of the ZTK to allow for better maintenance/release management/communication/marketing.
- Christian Theune: I'd like us to ponder how we can (in addition to the housekeeping and cleanups we do) also move to do constructive work together to expand the stuff that Zope packages (ZTK) is about. How do we go about implementing new technologies together, like supporting HTML 5 in the various parts? I'd like to start putting in new code in the foreseeable future in the zope.* namespace.

### Summary

The IRC log is available here: http://zope3.pov.lt/irclogs-zope/%23zope.2010-05-25.log.html#t2010-05-25T18:00:25

### Windows builds

Adam didn't manage to look into the AMIs due to illness. He did draft an initial proposal that he started discussed with Christian Theune to get funding for a hosted windows build machine from the foundation.

However, Amazon might not be the right choice due to the high cost associated with 64-bit machines. Rackspace's Windows VMs seem to be a viable alternative and will be investigated by Adam. Adam and Christian will write a proposal to present at the next board meeting on 2010-06-02 that:

- includes a trial period ( funding for 1-2 months to set everything up and make sure it does what we need)
- subsequent cost after the trial period
- ensures control over the machine is available to the foundation but can be handed over to individual community members easily

An open question is whether 64-bit Windows installations allow installing the build chains for 32-bit and 64-bit in parallel. (Adam to ask Sidnei for input)

### KGS 3.4.1 release

The KGS release is going to have a release candidate as soon as Adam gets to it. He noted that actual .tar and .exe releases will be made on request (the option will be stated in the release notes).

In an earlier meeting Adam wanted to write documentation about the release process but pointed out that there's already sufficient documentation in zope.release that helped him make the releases. Additional baijum and srichter pointed out resources (http://wiki.zope.org/zope3/DeveloperInfo#release-management and http://wiki.zope.org/zope3/MakingARelease) that are related to this.

Tres noted that we might put a new item on our list: generally consolidating the documentation floating around into the Sphinx documentation we have on docs.zope.org.

### Bugday review

Individual reports:

- Adam investigated Azure for building packages instead of using Amazon but that turned out to be a dead end. Azure is more like GAE than rented virtual machines.
- Christian Theune worked on emptying out the Zope 3 bugtracker and fixed some bugs along the way. There's 25 bugs left in the tracker now.
- Jens Vagelpohl worked on Zope 2 bugs and hit a dozen.
- Tres landed a patch from bzr and made releases of packages in the aftermath.
- Baijum wrote a few test cases for zope.mkzeoinstance.

Christian Theune noted that we should ponder how to establish a rule for releasing packages after bugs were fixed without making individual releases for every change. Having releases shortly after a bug day (like Tres did) seems reasonable.

We also would like to have some metrics that show what happened on a bug day. Various options were raised:

- note bugs that have been worked on in the wiki
- tag bugs with a unique tag
- use a query in LP for "everything that changed on day X"

The next bug day will be agreed upon with an open doodle to the Zope developers list, Christian will invite for the week that includes 2010-06-15.

### 2010-05-18

This is the agenda and summary for the weekly Zope developer meeting of Tuesday, 2010-05-18 on #zope@irc.freenode.org from 15:00 to 15:30 UTC.

### Agenda

- **Windows builds**
  - AMI (Sidnei, Adam)
  - Compiler licenses (Tres)
  - Amazon funding (Adam, Christian Theune)
- **Bug tracking**
  - Monitoring tracker status (Charlie Clark, ctheune)
- Preparing for Bugday 2010-05-19

### Ongoing issues

Those issues are currently ongoing. We don't have to discuss them. We just need to follow up on them eventually.

- **KGS 3.4.1 release**
  - Document release procedure (Adam)
  - Index view for download.zope.org
- **ZTK status**
  - **Towards a ZTK release**

        * Documentation
        * Release scope
- **Test runners / nightly builds**
    - Windows machines
- **Meta**
    - How to organize open issues in the long run (Blueprints? Other tool? Continue text files?)
    - Find second person to run the weekly meetings

### Topic proposals

- Lennart: Of course what applies to Hanno should apply to others making releases of packages maintained by the Zope Toolkit project as well. I think the ZTK leadership should figure out some kind of guidelines for this that people can follow.
- Chris McDonough: Pondering *some* (re-)structuring of the ZTK to allow for better maintenance/release management/communication/marketing.
- Christian Theune: I'd like us to ponder how we can (in addition to the housekeeping and cleanups we do) also move to do constructive work together to expand the stuff that Zope packages (ZTK) is about. How do we go about implementing new technologies together, like supporting HTML 5 in the various parts? I'd like to start putting in new code in the foreseeable future in the zope.* namespace.

### Summary

The IRC log is available here: http://zope3.pov.lt/irclogs-zope/%23zope.2010-05-18.log.html#t2010-05-18T18:07:10

### Windows builds

Adam didn't manage to look at the AMI builds yet, but he'll try for next week.

The VS licenses haven't turned up yet either and Christian Theune agreed to bring the issue up at the next foundation's board's meeting.

The foundation board agreed in general that the foundation will finance build infrastructure for Windows but needs a volunteer from the community to take responsibility on making a plan what to buy, how to maintain it, etc. Adam agreed to draft a plan and Christian Theune offered to help putting it into shape for presenting it to the board.

### Bug tracking

The Zope 3 bug tracker is almost emptied out [By the time of writing the protocol there's 25 open bugs left in it.]. We'll continue to migrate/triage the remaining bugs in the coming days/weeks.

The two project groups (zopetoolkit and zopeapp) provide a good overview of bugs in the official packages by now (see https://bugs.edge.launchpad.net/zopetoolkit and https://bugs.edge.launchpad.net/zopeapp) although the zopeapp list seems suspiciously short.

Christian Theune would like to get into a more steady mode of triaging them to lower the response time for people who report bugs. One part of this is the check-bugs script done by Charlie Clark which will be integrated into the daily test aggregation to report about languishing bugs. Christian Theune will set up the check script on one of his servers that also runs the repository checks.

Another part is to clean up the situation about bug notifications from Launchpad so that individual developers get notified in time. Christian Theune has been working on merging the various teams in Launchpad that exist so that there will be an administrative one (for managing the Launchpad project metadata) and one for developers (to handle bugs and day-to-day tasks). Once this is done individual developers should receive Launchpad notifications for newly reported bugs.

### Bug day 2010-05-19

The next bug day is this week on 2010-05-19. We didn't manage to talk much about this in preparation, although there was some triaging done by people and http://wiki.zope.org/ztk/BugDay20100519 shows a good list of people wanting to work on bugs.

### 2010-05-11

This is the summary of the weekly Zope developer meeting which happened on Tuesday, 2010-05-11 on #zope@irc.freenode.org from 15:00 to 15:30 UTC.

The agenda for this meeting is available in the mailing list archives: https://mail.zope.org/pipermail/zope-dev/2010-May/040543.html

The IRC log is here: http://zope3.pov.lt/irclogs-zope/%23zope.2010-05-11.log.html#t2010-05-11T18:03:06

### Present

Charlie Clark (convenor), Christian Theune, Christophe Combelles, Adam Groszer, Baiju Muthukadan, Sidnei da Silva

### KGS Status

3.4.1b1 has been released. Since last week all tests past and an announcement has been made to encourage testing as documentation is outstanding. Another release is planned this weekend.

Adam to coordinate releases with Jens. More information on release procedure or the steps required so that the KGS can be released directly would be appreciated.

Baiju suggested at least an index view for http://downloads.zope.com

### Windows Builds

Sidnei has made his Amazon image available to Adam for testing who will head this effort. It was agreed that Python 2.5 on Windows 64 will never be available for technical reasons.

The Zope Foundation has agreed to cover the costs for the buildbots but requires someone to be officially responsible for this. (Adam to coordinate with Christian Theune)

Still no feedback from Microsoft about the MSDN licences. From last week there still seemed to be some confusion about the validity of the licences for the proposed use. (Continue to badger Microsoft about this and while we're at it - has anyone asked them about using Azure for the Windows buildbots?)

### ZTK Status

There is agreement in the steering committee and progress. Christophe has a tool for detecting upgrades http://bitbucket.org/ccomb/z3c.checkversions which will go into the Zope repository at some point. (Check versions into repository and an issue tracker on Lauchpad)

### Bugday

A reminder to all that there will be a bugday next week and all are invited to sign up http://wiki.zope.org/ztk/BugDay20100519 Participation can be limited to reviewing bugs with patches.

Christian Theune hopes to do some triaging at the weekend, but there's more triaging required.

### Housekeeping

The minutes of all meetings will be archived at svn.zope.org/repos/main/zopetoolkit/doc/source as per Hanno's recent suggestion.

### 2010-05-04

This is the summary of the weekly Zope developer meeting which happened on Tuesday, 2010-05-04 on #zope@irc.freenode.org from15:00 to 15:45 UTC.

The agenda for this meeting is available in the mailing list archives: https://mail.zope.org/pipermail/zope-dev/2010-May/040493.html

The IRC log is here: http://zope3.pov.lt/irclogs-zope/%23zope.2010-05-04.log.html#t2010-05-04T18:00:13

### Present

Charlie Clark (convenor in Theuni's absence), Tres Seaver, Hanno Schlichting, Christophe Combelles, Lennart Regebro, Adam Groszer, Baiju Muthukadan, Marius Gediminas, Leonardo Almeida, Chris Warner, Fred Drake

### KGS Status

3.4.1 is nearly ready for release - all tests pass. Some housekeeping still required for a beta release.

- Adam to check with Jens Vagelpohl on getting access to dowloads.zope.org

### Test runners/nightly builds - Windows machines

- 2 test failures on Python 2.4 & 2.5 from a total of 9404 tests
- Still waiting to hear from Microsoft regarding the licences (Tres will follow this up)
- Amazon image still required from Sidnei
- Grok smoketest required (To be taken up by the ZTK release team due to meet on Thursday)

### Deprecating older Python versions

4:1 votes in favour of officially dropping support for Python 2.4.

However, after an impassioned plea by Fred Drake it was agreed that providing legacy support for Python 2.4 in ZTK 1.0.x would cost little. Problems are currently restricted to doctests and workarounds are known. No one should start new ZTK-based work with Python 2.4

Support will be dropped completely from ZTK 1.1 on and in any case by 2010-12-31. ZTK 1.1 will allow features that are not compatible with Python 2.4 and possibly also with 2.5

### Bug day reminder

The next bug day is 2010-05-19.

Launchpad gardening and triaging will be required before then.

### 2010-04-06

This is the summary of the weekly Zope developer meeting which happened on Tuesday, 2010-04-06 on #zope@irc.freenode.org from 3pm to 3:30pm (UTC).

The agenda for this meeting is available in the mailing list archives: https://mail.zope.org/pipermail/zope-dev/2010-April/039981.html

The IRC logs are located here: http://zope3.pov.lt/irclogs-zope/%23zope.2010-04-06.log.html#t2010-04-06T18:00:01

Tres Seaver was filling in for Christian Theune as conventer due to illness.

### Zope2 release manager

The community welcomes Hanno Schlichting as the new release manager for Zope2. In addition to closing / responding to dozens of bugs in the past week, Hanno has also proposed a roadmap for Zope 2.13 and has made the release tag for Zope 2.12.4.

### Porting packages to Zope3

Lennart Regebro reported that he has branches waiting for the first three "most depended-on" ZTK packages (`zope.event`, `zope.interface`, and `zope.testing`). After considering naming the new releases `4.0`, the consensus was to just name them the next "normal" major release, as long as there are no backward-incompatible API changes.

Some discussion of the general problem of doctests breaking due to exception formatting led to a suggestion from Jim Fulton that we implement a testing API similar the `unittest.TestCase.assertRaises`, but with the additional feature that it returns the exception value, to permit further assertions about the state of that object.

### Buildbots

No one had new updates on the topic. Sidnei da Silva reported that he is close to reconstructing the script used to build `win64` images for Amazon EC.

Tres Seaver reported that his request to Microsoft for dontaed VisualStudio licnese to support builds / tests of `zope.*` packages on Windows for Python 2.4 and 2.5 is still stalled.

Baiju M noted that Rackspace can also support Windows in its cloud: http://www.rackspacecloud.com/ .

### Tracker status

Charlie Clark reported that he had gotten his script for checking Launchpad for "languishing" bugs working. He posted that package to `zope-dev@zope.org` during the meeting.

### Bug Days

After some discussion, Adam Groszer and Baiju M agreed to widen their planned `bluebream` sprint to include more general ZTK issues. The sprint, originally set for Saturday, 2010-04-10, was rescheduled to Saturday, 2010-04-24. Charlie Clark asked for announcments, plus volunteers willing to help mentor new exterminators during the bug day. Adam and Baiju will organize a web page for coordinating the bug day activity, and send out the announcements.

### Carried Over

- Resurrecting the "sprint schedule" page, http://wiki.zope.org/zope3/SprintSchedule
- A more general calendar (e.g., for Zope / Python related conferences, symposia, sprints, etc.)

## Release team

The ZTK release team oversaw the release process of the Zope Toolkit. It communicated on the zope-dev mailing list (*Mailing list*).

### Members

- Christophe Combelles (representative for BlueBream)
- Jan-Wijbrand Kolman (representative for Grok)
- Hanno Schlichting (representative for Zope)

### ZTK meeting - 2010-05-06

### Attendance

ccomb, j-w, hannosch

### Agenda

- No fixed agenda, this is a kick-off meeting.

### Discussion

Communication

- We'll use the zope-dev mailing list for our discussions and no separate list

Our role

- We see ourselves as representatives of communities that make use of the ZTK
- We should ensure stable releases of the ZTK, which are useful to our projects not more and not less

Release outcome

- Should produce a http://download.zope.org/ztk/release/1.1 with a ztk.cfg in it and a the zopeapp.cfg (for as long as it exists) in it.
- Nice to have: an index (for easy_install people)
- Should have some documentation site stating changes (http://docs.zope.org/zopetoolkit/releases/overview-trunk.html)

Release policies

- At first manual releases (x.y.Z), automate the process to generate the bugfix releases later. We need to make sure to release only versions sets for which all tests passed.
- ztk x.y.z. releases, stable package list per release
- a ztk minor release per month would be ok
- a ztk major release "when one of the consumers projects needs it".
- backward compatibility breaking only happens in X.y.z that means, if we have a zope.component 4.0.0, it will be part of ZTK 2.0 or 3.0
- generally upstream releases happen in a 6-12 month interval, so the same timeline makes sense for ZTK releases

### Tasks

- We want 64bit Linux and Windows tests for the ZTK. j-w is bugging janjaap to create those. Maybe contact ccomb for adding slaves to "afpy" [j-w]
- Make sure we have a buildbot testing the ZTK releases (and not SVN) [ccomb]

### Open points

- Look at Tres's list of packages in all three frameworks, decide on a way to drop things from the initial ZTK set and on a process for the future.
- Look at and update http://docs.zope.org/zopetoolkit/about/coreextra.html for adding/removing policies, probably have a deprecated.cfg file
- Decide on process for new feature versions and the process for going from 1.1.0 alpha to a final

### Next meeting

2010-05-18, 14:00 to 15:00 UTC before the zope-dev meeting, in #zope

### ZTK meeting - 2010-05-18

#### Attendance

ccomb, j-w, hannosch

#### Agenda

- Minutes of last meeting
- Follow-up on tasks, open points
- New and ongoging tasks, open points.
- Planning the next meeting

#### Discussion

Minutes of last meeting

- The minues were principly agreed on.
- The minutes of the ztk-release-team meeting are located at http://svn.zope.org/zopetoolkit/doc/source/releaseteam/

Buildslaves testing the ZTK on required platforms

- There have been a number of buildbot-slaves setup already. The reports of these slaves should be sent to zope-test. The mail-subject needs to be foramtted propely for this to work. j-w will coordinate with Jan-Jaap.
- We still need slaves for 64-bit Windows. Work is underway to obtain the correct Windows licenses.
- ccomb has setup a buildbot for testing ZTK against released packages (not SVN) on 32-bit linux. He will wake it up and have the reports send to zope- test as well.

Deprecating packages from ztk.cfg and zopeapp.cfg

- There will be a deprecating.cfg where packages are moved from ztk.cfg and zopeapp.cfg.
- The "under-review" and "unresolved-dependencies" sections need special scrutiny.
- Tres Seaver posted a matrix depicting what ZTK packages are in use by what frameworks. Each of the represenatives will review the list of packages all three frameworks use. When putting the results together, this should result in a cleaned up ztk.cfg, a cleanup up zopeapp.cfg and a deprecating.cfg. [ccomb, hannosch, j-w]

ZTK-1.0

- The release team will do an initial pass on splitting the package lists into ztk.cfg, zopeapp.cfg and deprecating.cfg to come to a stable set.
- After this first round, we will declare the ZTK 1.0a1 to have a concrete starting point.
- The ZTK-1.0 will be used from Grok-1.2 and BlueBream-1.1 on. The upcoming Grok-1.1 release will refer to a specific, slightly older, ZTK revision from SVN. Work is going on on Grok-1.2 already and is based on a very recent revision of the ZTK.
- hannosch has access to http://download.zope.org/ztk/release/ for uploading the releases.

### Tasks

- We want 64bit Linux and Windows tests for the ZTK. j-w is bugging janjaap to create those. Maybe contact ccomb for adding slaves to "afpy". We can resolve this issue when the build reports are comming in on zope-test. [j-w]
- Make sure we have a buildbot testing the ZTK releases (and not SVN) [ccomb]
- Each of the represenatives will review the list of packages all three frameworks use, as made by Tres. [ccomb, hannosch, j-w]

### Open points

- Look at and update http://docs.zope.org/zopetoolkit/about/coreextra.html for adding/removing policies, probably have a deprecated.cfg file
- Decide on process for new feature versions and the process for going from 1.1.0 alpha to a final
- the list for "extra" packages still needs to be defined, we could have a incoming.cfg and/or extra.cfg

### Next meeting

2010-05-25, 14:00 to 15:00 UTC before the zope-dev meeting, in #zope

### ZTK meeting - 2010-06-01

### Attendance

ccomb, j-w, hannosch

### Agenda

- Minutes of last meeting
- Follow-up on tasks, open points
- New and ongoing tasks, open points.
- Planning the next meeting

### Discussion

Minutes of last meeting

- The minutes were principally agreed on.

Buildslaves testing the ZTK on required platforms

- The test reports from the buildslaves as setup by Jan-Jaap testing the ZTK on the Windows platform are now indeed sent to zope-test.
- We still need 64-bit Windows slaves. Jan-Jaap notes that The Health Agency will be somewhat busy the coming period, so these slaves might need to wait a little more. Unless of course someone else sets up slaves.

- Note that the buildslaves set up by THA test the‘‘released‘‘ packages and thus should not need a compiler installed. So, an MSDN license is not necessary for these slaves.
- Ccomb made sure we have a buildbot testing the ZTK releases (and not SVN).

ZTK-layout

- ccomb started a ZTK branch where he made a new layout for the ZTK files.
- ccomb suggested creating and maintaining an `extras.cfg` where packages that make up “a common ecosphere” of software would be tested with and against the official ZTK.
- hannosch and j-w didn’t see the need for such an `extras.cfg` as it would put extra maintenance burden on the ZTK-release-team efforts without a clear benefit. They’re in favor of a clearly defined set of packages that define the essence of a ZTK and nothing more.
- After discussions a decision was made to remove the concept of the `extras.cfg` and go from there. If at some point in the future we think such a list of extra package is needed after all, we can resurrect the concept.
- Since most of the ZTK-consumer frameworks will actually maintain lists of extra packages needed for their frameworks, it was suggested to send in automated test reports about the respective ecospheres. This will provide a sense of coherence and stability. (see for example the “Grok Toolkit” that extends the ZTK).
- Breakage of these ecospheres will be the responsibility of the respective communities.
- The ZTK contains sections referring to unresolved dependencies and other questionable packages. These packages need to be reviewed.

### Tasks

- We want 64bit Linux and Windows tests for the ZTK. j-w is bugging janjaap to create those. Maybe contact ccomb for adding slaves to “afpy”. We can resolve this issue when the build reports are coming in on zope-test. [j-w]
- Each of the representatives will review the list of packages all three frameworks use, as made by Tres. [ccomb, hannosch, j-w]
- Update the http://docs.zope.org/zopetoolkit/about/coreextra.html documentation. [hannosch]
- Update the ccomb branch to reflect the `extras.cfg` decision. [ccomb]
- Review the unresolved dependencies sections in the ZTK. [j-w]

### Open points

- Decide on process for new feature versions and the process for going from 1.1.0 alpha to a final.

### Next meeting

2010-06-15, 20:00 to 21:00 CET on #zope

### ZTK meeting - 2010-06-15

### Attendance

ccomb, j-w, hannosch

### Agenda

- Minutes of last meeting
- Follow-up on tasks, open points
- New and ongoing tasks, open points.
- Planning the next meeting

### Discussion

Minutes of last meeting

- The minutes were principally agreed on.

ZTK-layout

- Hanno updated http://docs.zope.org/zopetoolkit/about/libraries.html to reflect the ztk release team's vision.
- zope.file and zope.html will be removed from the ztk as they do not seem fundamental to the toolkit.
- Work will continue for the other "under-review" packages.
- Work will continue on implementing the decided packages policy in the ZTK.
- Zope2's next release will be an alpha next week. The final is target somewhere in september/october. Conceptually this new Zope2 release will be based on ZTK 1.0.

### Tasks

- Update the ccomb branch to reflect the `extras.cfg` decision. [ccomb]
- Review the unresolved dependencies sections in the ZTK. [j-w]
- See about the 64-bit windows ztk buildbot slaves. [j-w]
- Merge the "ccomb" branch. [hannosch and ccomb]

### Open points

- Decide on process for new feature versions and the process for going from 1.1.0 alpha to a final.
- The upgrade versions buildbot updates to non-final releases, it shouldn't. Example: zc.recipe.egg=1.2.3b2 # was: 1.2.2 [ccomb?]
- Make an alpha release and document the release procedure.
- Tres Seaver posted a matrix depicting what ZTK packages are in use by what frameworks. Each of the representatives will review the list of packages all three frameworks use. When putting the results together, this should result in a cleaned up ztk.cfg and zopeapp.cfg. [ccomb, hannosch, j-w]

### Next meeting

Ideally somewhere in the week of 2010-06-22. Team members will keep each other updated during the coming week.

### ZTK meeting - 2010-06-30

#### Attendance

j-w, hannosch (ccomb joined in later).

#### Agenda

- Minutes of last meeting
- Follow-up on tasks, open points
- New and ongoing tasks, open points.
- Planning the next meeting

#### Discussion

Minutes of last meeting

- The minutes of the last meeting were principally agreed on.

ZTK alpha release.

- The merge of ccomb's ZTK branch is done. The ZTK structure is now in place. Hanno has created a test-release to document the release process. This went well and he is about to make an official alpha release!
- The "unresolved" and "deprecated" dependencies in the ZTK has been reviewed. All of the packages in these two lists have been removed from the ZTK.
- There was a long discussion about the usefulness of a custom ZTK package index besides the `*.cfg` files that depict the packages versions. Buildout- based projects do not require a custom index, only the version files. "Easy- install"-based projects though will need a custom index to control the versions of the packages that get installed.

  However, it was questioned whether the custom index should contain packages that are not managed as part of the ZTK, but still are dependencies of the ZTK (ZODB is such a prominent example) as it would impose too much version- policy on consumers of the index.

  The releaseteam is not sure this is actually the case, but, since the members of the releaseteam all represent buildout-based projects, the releaseteam cannot really know for sure either. It was concluded not to include a custom index as part of a ZTK release for the moment. Feedback concerning this issue (or maybe even a representative) from a "easy-install"-based ZTK- consumer-project is very much welcome.

#### Tasks

- See about the 64-bit windows ztk buildbot slaves. [j-w]
- Talk to Adam (Grozer?) about 64-bits windows ZTk test machine [hannosch]
- Test Grok-1.1 and trunk against ZTK alpha releaase and report [j-w]
- Test BB against ZTK alpha release and report [ccomb]

### Open points

- Decide on process for new feature versions and the process for going from 1.1.0 alpha to a final.
- The upgrade versions buildbot updates to non-final releases, it shouldn't. Example: zc.recipe.egg=1.2.3b2 # was: 1.2.2 [ccomb?]
- Tres Seaver posted a matrix depicting what ZTK packages are in use by what frameworks. Each of the representatives will review the list of packages all three frameworks use. When putting the results together, this should result in a cleaned up ztk.cfg and zopeapp.cfg. [ccomb, hannosch, j-w]

### Next meeting

Next meeting is scheduled for July 06 16:00 (CET) on #zope.

## ZTK meeting - 2010-07-06

### Attendance

j-w, hannosch, ccomb.

### Agenda

- Minutes of last meeting
- Follow-up on tasks, open points
- New and ongoing tasks, open points.
- Planning the next meeting

### Discussion

Minutes of last meeting

- The minutes of the last meeting were principally agreed on.

ZTK alpha release.

- Not much feedback from "direct consumers", Hanno will ask for in direct communication with some of the known "consumers".
- Grok works with the released alpha(!).
- Both Grok and Zope2 use their own "toolkits", that are extending the ZTK. Bluebream does not (yet) have a "bbkit".
- The roadmap for the 1.0 release is: 1.0b1 around August 15th, 1.0 final in September. Dates will be more precise after the holidays of ccomb and j-w.

### Tasks

- Checkversions script will be updated [ccomb]
- Add windows slaves to overview of buildslaves [hannosch]
- Test BB against ZTK alpha release and report [ccomb]
- The upgrade versions buildbot updates to non-final releases, it shouldn't. Example: zc.recipe.egg=1.2.3b2 # was: 1.2.2 [ccomb?]
- Tres Seaver posted a matrix depicting what ZTK packages are in use by what frameworks. Each of the representatives will review the list of packages all three frameworks use. When putting the results together, this should result in a cleaned up ztk.cfg and zopeapp.cfg. [ccomb, hannosch, j-w]

### Open points

- Decide on process for new feature versions and the process for going from 1.1.0 alpha to a final.

### Next meeting

Next meeting is scheduled for July 13 16:00 (CET) on #zope.

### ZTK meeting - 2010-08-17

### Attendance

j-w, hannosch, ccomb.

### Agenda

- Minutes of last meeting
- Follow-up on tasks, open points
- New and ongoing tasks, open points
- Planning the next meeting

### Discussion

Minutes of last meeting

- The minutes of the last meeting were principally agreed on.

Tasks

- BB-1.0b3 uses the ZTK release.
- The checkversions script includes "prefer-final" now.
- http://buildbot.afpy.org/discover-ztk1.0/waterfall is a buildbot that will continuously build a working list and a blacklist.
- Windows slaves have been added to the buildslaves overview.

Deprecation of packages in the ZTK

- We'll deprecate every zope.app.* package not used by any framework according to http://docs.zope.org/zopetoolkit/releaseteam/package-usage.html from the ZTK.
- We'll deprecate zope.documenttemplate from the ZTK.
- We'll consider renaming zope.app.wsgi to zope.wsgi.
- We'll need to investigate to split up zope.app.appsetup into zope.appsetup and Grok- and BB-specifics. zope.appsetup could then be part of the ZTK.
- We'll leave zope.server in the ZTK for 1.0 and reconsider for 1.1.
- There's still a series of test failures on the Windows platform. A 1.0 release should see all tests pass on all chosen platforms though. This requires some work to be done.

### Tasks

- Add a remark about the discover-ztk buildbot to the release-procedures.rst file [ccomb].
- Change to `*.cfg`'s for deprecating the packages [hannosch].
- Request download.zope.org access from Jens Vagepohl [j-w].
- Next alpha release planned for 08/23 16:00 (CEST) [hannosch, j-w].

### Open points

- Will Windows test failures for the ZTK block a beta?

### Next meeting

Next meeting is scheduled for August 24 16:00 (CEST) on #zope.

### ZTK meeting - 2010-08-24

### Attendance

j-w, hannosch, ccomb.

### Agenda

- Minutes of last meeting
- Follow-up on tasks, open points
- New and ongoing tasks, open points
- Planning the next meeting

### Discussion

Minutes of last meeting

- The minutes of the last meeting were principally agreed on.

Tasks

- hannosch updated the *.cfg's deprecating zope.app.* packages that are not used by the consuming frameworks.
- j-w got access to download.zope.org for making ZTK releases. He will do the next alpha release just after this meeting.
- There're still Windows-related test failures for the ZTK. Work still needs to be done to identify and fix these issues.
- ccomb brought up the issue of sharing a list of packages between BB and Grok that will not be part of the ZTK. The suggestion is to start collaborating on such a list between the respective communities.

### Tasks

- Make the 1.0a3 release. [j-w]
- Windows test failures [j-w]
- Add a remark about the discover-ztk buildbot to the release-procedures.rst file [ccomb].

### Open points

- Will Windows test failures for the ZTK block a beta?

### Next meeting

Next meeting is scheduled for September 7th 16:00 (CEST) on #zope.

### ZTK meeting - 2010-09-07

### Attendance

j-w, hannosch, ccomb.

### Agenda

- Minutes of last meeting
- Follow-up on tasks, open points
- New and ongoing tasks, open points
- Planning the next meeting

### Discussion

Minutes of last meeting

- The minutes of the last meeting were not finished in time.

Tasks

- The 1.0a3 release was made just after the last meeting. Thusfar the releaseteam did not get negative feedback on this release.
- There are not too many Windows test failures left and all had to do with support for Python-2.4.4.
- The documentation status for the packages of the ZTK was discussed. They should be easily accessible. They are now on http://docs.zope.org/ztkpackages.html and http://docs.zope.org/ztkapppackages.html. For the 1.0 release the releaseteam deems the current status as sufficient. Documentation quality could affect a package's status within the ZTK.
- The landing page on http://download.zope.org/zopetoolkit/ should provide helpful pointers to more information.
- We'd like to have an aggregated changelog for the ZTK packages. The BB project has some scripts that maybe can be reused.
- hannosch and j-w will attend the Zope Summit on September 13th and 14th @gocept.

### Tasks

- Landing page for http://download.zope.org/zopetoolkit/. [hannosch]
- Windows test failures [j-w]
- Add a remark about the discover-ztk buildbot to the release-procedures.rst file [ccomb].

### Open points

- ...

### Next meeting

Next meeting is scheduled for September 21st 16:00 (CEST) on #zope.

### ZTK meeting - 2010-09-21

### Attendance

j-w, hannosch, ccomb.

### Agenda

- Minutes of last meeting
- Follow-up on tasks, open points
- New and ongoing tasks, open points

- Planning the next meeting

### Discussion

Minutes of last meeting

- The minutes of the last meeting were not finished in time.

Tasks

- Hannosch will try to work on a landing page for download.zope.org.
- Hannosch, with the help of icemac, have been working on fixing test dependencies for the ZTK 1.0. There is some work left. Of specific interest is the zope.app.interface package that BB currently depends on.
- There's a lack of overview over the various buildbots that test the ZTK (in various profiles). It was suggested to add an *svn info* step to the builds. This will provide enough information for reproducing (failed) builds in order to effectively fix issues. Agrozer volunteered to ask the various buildbot maintainers to add such a build step.
- In these cases where a package breaks the ZTK builds, a maintenance branch for that package will be created.
- Once the ZTK 1.0 released and work for 1.1 is undertaken, even more of the zope.app.* packages will be deprecated. Especially in the area of test dependencies. It was noted that zope.component.testlayer, zope.app.wsgi.testlayer and zope.app.appsetup.testlayer have test infrastructure that could help migrating away from zope.app.testing and zope.app.zcmlfiles. These testlayer modules were created by Martijn Faassen and Sylvian Viollon during the two "zapp zope.app" Grok sprints.

### Tasks

- Landing page for http://download.zope.org/zopetoolkit/. [hannosch]
- Setting up new ZTK "giant" buildbot [ccomb].

### Open points

- ...

### Next meeting

Next meeting is scheduled for September 28st 16:00 (CEST) on #zope.

### ZTK meeting - 2010-11-09

### Attendance

j-w, hannosch, ccomb.

### Agenda

- Discussion
- Tasks

- Planning the next meeting

### Discussion

Grok-1.2 has been released, based on ZTK-1.0. Grok still depends on a handful of "zope.app.*" packages. J-w will list them. Except for these, all other zope.app.* package will be deprecated for ZTK-1.1.

BlueBream is using ZTK-1.0 and only short of an 1.0 release.

zope.app.authentication will be deprecated, and the latest zope.pluggableauth included in the ZTK.

ZTK-1.1 is supposed to run on Python-2.7, Python-2.4 support is dropped. This means, the ZODB version can be updated to 3.10.x. There are actually many more package that can now be updated in the ZTK, like zope.testing and zope.testrunner.

The ZTK would like to use Distribute by default.

The buildout.cfg files in maintenance branches of packages in the ZTK should extend from a specific version of the ZTK themselves to get a predictable result.

The trunks of packages in the ZTK could, for example like the Grok project does, extend from the `*.cfg` files in the ZTK trunk. Specific version pins can be defined in the package's [version] section.

There needs to be a roadmap for Python-3.0 support. Hanno will inform with Lennart Regebro about the current status.

Hanno will check for minor version updates in the packages and see if it warrants a ZTK-1.0.1 release.

### Tasks

- update minor version of packages for ZTK-1.0.1 [hannosch]
- deprecate zope.app.* from ZTK trunk [j-w]
- update all packages in the ZTK trunk [ccomb]

### Next meeting

Next meeting is scheduled for November 23rd 20:30 (CET) on #zope.

## ZTK meeting - 2010-11-23

### Attendance

j-w, hannosch, ccomb.

### Agenda

- Discussion
- Tasks
- Planning the next meeting

### Discussion

- j-w moved the *zope.app.** packages to the deprecated-section.
- There was a short discussion on zope-dev about using distribute versus setuptools. The releaseteam conclusion was not to introduce any (implicit) dependencies on either. Hanno suggested to test the ZTK packages for (unintentional) dependencies on distribute.
- There're a few ZTK packages trunks that refer to yet-unreleased versions of their dependencies. This means, these trunk a hard to release, but more importantly, these packages will not be easily testable in buildbots.

  Various options were discussed including: using mr.developer and its auto-checkout features, refrain from testing trunks in buildbot or have developments done on a packages that requires a new release of a dependency, done on development branches. This latter seems to be the sensible approach.

  J-w will make a list of packages trunks that have this issue.

### Tasks

- Check for ZTK packages that would rely on distribute [hannosch]
- Check for package trunks that rely on yet unreleased version of its dependencies [j-w]
- Update the version in the ZTK [ccomb]

### Next meeting

Next meeting is scheduled for December 7th 20:30 (CET) on #zope.

## ZTK meeting - 2010-12-07

### Attendance

j-w, hannosch.

### Agenda

- Discussion
- Tasks
- Planning the next meeting

### Discussion

The tasks of last meeting were not handled yet. Since no new issues came up in the meantime and ccomb was not able to make it to the meeting in time, it was decided to adjourn the meeting until December 14th.

### Tasks

- Check for ZTK packages that would rely on distribute [hannosch]
- Check for package trunks that rely on yet unreleased version of its dependencies [j-w]
- Update the version in the ZTK [ccomb]

### Next meeting

Next meeting is scheduled for December 14th 20:30 (CET) on #zope.

## ZTK meeting - 2010-12-14

### Attendance

j-w, hannosch.

### Agenda

- Discussion
- Tasks
- Planning the next meeting

### Discussion

- Hannosch tested the ZTK packages for unintentional dependencies on distribute versus setuptools. His conclusion was there is no such dependency at this point. This is good.
- Several package trunks relied on un-released versions of other packages. Hannosch released several packages to resolve this situation.
- z3c.recipe.compattest needs a release to fix its inability to run in a virtualenv. This will be taken care of by Jan-Jaap Driessen.

### Tasks

n/a

### Next meeting

Next meeting is scheduled for January 4th 20:30 (CET) on #zope.

## ZTK meeting - 2011-01-04

### Attendance

j-w, ccomb, hannosch.

### Agenda

- Discussion
- Tasks
- Planning the next meeting

### Discussion

- ccomb plans the release BlueBream 1.0 "final" soon.
- z3c.recipe.compattest has been released and runs again in a virtualenv.
- A planning is made for the ZTK-1.1 release. Since a Grok release is on the horizon, it makes sense to do a ZTK-1.1 next week. j-w will announce this plan to zope-dev. hannosch will write a high-level changelog for ZTK-1.1.
- Since a lot of zope.app.* packages have been deprecated from the ZTK, the ones that are still part of the ZTK are discussed and zope.app.appsetup in particular. Both BB and Grok use this package, even though Grok is investigating how to move the policy defined in zope.app.appsetup to the Grok codebase.
- The plan was crystalized to indeed do a ZTK-1.1 release on Tuesday January 11th.

### Tasks

- hannosch: write a high-level changelog for ZTK-1.1
- j-w: short announcement to zope-dev about the ZTK-1.1 release plans.

### Next meeting

Next meeting is scheduled for January 11th 20:30 (CET) on #zope.

## Steering Group

The formation of the Zope Toolkit was facilitated by the Zope Toolkit Steering Group. It was influential in defining the scope of the ZTK and establishing community practices and policies around it.

In early 2010 the steering group members stepped down, after succeeding in the creation of the toolkit concept and getting wide community support for it.

After the steering group members stepped down, the Zope community agreed on the formation of the release team (*Release team*) with a much narrower focus around release management of the ZTK.

### How we can help you

If you want to make some change to the Zope Toolkit, start a thread on zope-dev about it. The Steering Group is watching.

The Steering Group is going to make decisions in the following way:

- A steering group member participates in discussions as normally, saying +1 or -1 or whatever alternative suggestions they may have.

- If it's clear there's a consensus by the steering group, we will record the decision in a document in SVN to be announced. If not all steering group members voted and it's "enough time" later, fine, the decision is made by those who did care to contribute.
- If there is no consensus by the steering group, the most votes by steering group members in the thread win. It is the task of the steering group to detect this and record it (or delegate its recording by someone else, even better).
- If for some reason it's hard to count votes and get consensus in the Steering Group, people can point this out. "Steering Group help, no consensus detected!". We will then reach a consensus.
- If there's still no consensus, Martijn is going to flip a coin and that's going to be the answer. Martijn going to do his best to avoid having to flip any coins, as it's very silly to decide on the flip of a coin.
- If you are doing a larger project of changing the code (as opposed to minor changes or bugfixes), you should make a proposal on the mailing list and get a steering group member to participate in the discussion.
- If it's a small topic and no steering group member participates in a discussion and consensus is reached, please proceed. Everything's fine.
- So, you only need to worry about the steering group for relatively large issues, and when they're actually participating in a discussion. You can also invoke the steering group if you don't like the consensus reached in a discussion or if no consensus is reached.
- Feel free to call in the Steering Group and it will fly in. It's not a bird, it's not a plane, it's the Steering Group!

### Decisions

This is a temporary place to note decisions. The idea is to later integrate them into the Zope Toolkit documentation, but we need a quick way to note decisions first.

- ZMI related code shall not be part of the Zope Toolkit, and we shall strive to remove it.
- If zope.deferredimport is used in a package merely to avoid the use of `from` imports, then instead we will use `from` imports to get rid of this dependency.
- Files used to support the old ZPKG system such as `DEPENDENCIES.cfg` can be safely removed from packages.
- So-called "ZCML-slugs" which were intended to be symlinked into a special slugs directory in a Zope installation are not in use anymore. They should be removed. Typically they look like `zope.foo-configure.zcml`.
- we are going to work at getting rid of the zope.app.testing extra by distributing its facilities into individual zope.* packages. Hopefully we can get a clear consensus on this one from the people who object to the general policy.
- if you think a new "extra" dependency is needed in a Zope Toolkit package, and you're not just moving stuff between packages but actually developing new code, bring it up on zope-dev so we can at least consider alternatives. Perhaps a better home can be found for this code.
- We can consider removing extra dependencies for particular Zope Toolkit packages in order to make the dependency graph easier to reason about. We will do this on a case by case basis though.
- In namespace package's `__init__.py` we have been using the following boilerplate code:

```
try:
    import pkg_resources
    pkg_resources.declare_namespace(__name__)
except ImportError:
    import pkgutil
    __path__ = pkgutil.extend_path(__path__, __name__)
```

Since `setuptools` is a dependency of our packages anyway, we can instead do the following:

```
__import__('pkg_resources').declare_namespace(__name__)
```

Feel free to use that and also feel free to simplify existing `__init__.py` modules. Just make sure `setuptools` is a declared dependency of the package.

- Moving code around as part of dependency refactoring is worth a feature release (x.y as opposed to x.y.z version number) for the affected packages. Changing an import to make use of a new package that came out of such refactoring is also worth a feature release.
- If a package A starts to rely on new features in dependency B, that is worth a feature release of package A.
- The version requirements in setup.py should specify only API compatibility. They should not specify a dependency on bug fixes; that's the domain of the KGS.

  Therefore in a `setup.py` you are allowed to write this in `setup.py`:

  ```
  bar >= x.y
  ```

  I.e. relying on a newer feature release of package `bar`.

  but not this:

  ```
  bar >= x.y.z
  ```

  I.e. you're not allowed to rely on a newer *bugfix* release of package `bar`.

  Elaboration: Imagine package `foo` that depends on package `bar`. If you make changes in `foo` so that it starts to rely on changes in `bar` that are only introduced in a feature release of `bar` (`bar` version `x.y` or `bar` version `x`), you should update the `setup.py` of `foo` to state this dependency with a requirement like this:

  ```
  bar >= x.y
  ```

  This is only relevant to *feature* releases. If there is a bugfix release of `bar` you should not write a dependency like `bar >= x.y.z`.

  This is a compromise in the interests of both flexibility and giving hints to people who use our packages. We'll see how it goes.

- Some Zope Toolkit packages are quite reusable without having to buy into the rest of the Zope Toolkit. Others aren't reusable at all without pulling in a huge chunk of the Zope Toolkit; they depend on many assumptions.

  We should communicate this clearly to potential users. As a first step, we will make sure these notifications are available on the PyPI pages. We will do this by adding a message about reusability to the long_description (which gets displayed on PyPI). Typically this is done by modifying the package's README.txt or `src/zope/../README.txt` doctest.

  The following text should be used for reusable packages:

  ```
  *This package is intended to be independently reusable in any Python
  project. It is maintained by the* `Zope Toolkit project
  <http://docs.zope.org/zopetoolkit/>`_.
  ```

  The following text should be used for packages that are *not* easily reusable:

  ```
  *This package is at present not reusable without depending on a
  large chunk of the Zope Toolkit and its assumptions. It is
  maintained by the* `Zope Toolkit project
  <http://docs.zope.org/zopetoolkit/>`_.
  ```

At the time of writing, most of our packages will be marked as *not* reusable. Only packages at the roots of our dependency tree that have a clear purpose and some documentation (such as `zope.interface` and `zope.component`) should be marked as reusable. We will slowly start to build up from there.

- When code moves to a new location to import it from (in the same or another package), use a `from foo import bar` statement, with a `#BBB` comment to indicate the import is only there to support backwards compatibility.

  In the CHANGES.txt of a package, state that an import location got deprecated and where the new location is (making this a feature release, not a bugfix release).

  Reasons:

  - it avoid a dependency on zope.deprecation, which is quite involved in its implementation, using proxies.
  - A `from .. import ..` is immediately comprehensible to any Python programmer as well as tools.
  - Deprecation warnings make it hard to write a library that supports multiple versions of another library; a change in an indirect dependency can create deprecation warnings that the original developer does not care about.
  - We are in the process of developing a testrunner extension that will report on indirect imports, and a ZODB upgrade procedure.

- The open issues will be moved to the launchpad blueprints and launchpad answers. The blueprint specifications will be stored in the ZTK documentation and linked to (each blueprint will be a separate document).

- As a general direction we'd like to separate out the XML-RPC related code and FTP-related code into seperate packages that aren't depended on by the rest of the toolkit. This makes it possible for developers to use the toolkit without worrying about XML-RPC or FTP.

- To make it easy for developers to identify which packages are in the ZTK, as they are sometimes working in isolation, a package that is maintained by the Zope Toolkit project should have the following text in its `setup.py` file just under the copyright header:

```
##############################################################################
# This package is developed by the Zope Toolkit project, documented here:
# http://docs.zope.org/zopetoolkit
# When developing and releasing this package, please follow the documented
# Zope Toolkit policies as described by this documentation.
##############################################################################
```

  When a package is *removed* from the ZTK this header should be removed from its `setup.py` as well.

- There was a discussion about whether the version in `setup.py` should be set to `0` instead of the next expected version. There was a discussion and the majority of the steering group was against this change - the ZTK release policy remains the same for the ZTK.

  The mailing list threads discusses various pros and cons:

```
https://mail.zope.org/pipermail/zope-dev/2009-September/037725.html

https://mail.zope.org/pipermail/zope-dev/2009-September/037735.html
```

- We want to encourage narrative documentation for packages (doctested or not). Even though we do not require that this documentation is executable (doctested), we do prefer a project setup so that it easy to support executable documentation as well, in cooperation with tools like manuel. The bobo documentation is an example of such a setup. We should document this better.

### Members

Past members of the steering group:

- Martijn Faassen
- Stephan Richter
- Christian Theune <ct@gocept.com>
- Jim Fulton