# CONTENTS

1 **Installation** 3  
1.1 Installation ......................................................... 3  

2 **Getting Started: Users** 5  
2.1 Datapkg Manual ..................................................... 5  
2.2 Using datapkg from code using the API ......................... 5  
2.3 More information on specific features .......................... 5  

3 **Getting Started: Developers** 7  
3.1 Datapkg Design ..................................................... 7  
3.2 Use cases for datapkg .............................................. 8  
3.3 Research ........................................................... 10  
3.4 Extending Datapkg .................................................. 11  

4 **Other material** 13  
4.1 License ............................................................. 13  
4.2 History ............................................................ 13  
4.3 Notes on External Libraries ..................................... 16  

5 **Indices and tables** 19
datapkg is a tool for distributing, discovering and installing data ‘packages’.

datapkg is a simple way to ‘package’ data building on existing packaging tools developed for code. datapkg is designed to integrate closely with the CKAN (Comprehensive Knowledge Archive Network).

By putting data in a package, it gets labelled with standardized metadata and can be put in a datapkg repository, such as CKAN or a local one. Once in such a repository, the packages are easy to find and retrieve.
1.1 Installation

1. Install python: http://www.python.org/
2. Install setuptools library (and (optional) virtualenv)
3. Install datapkg
   Using setuptools’s easy_install:
   
   $ easy_install datapkg
   
   Or to install with pip (http://pypi.python.org/pypi/pip):
   
   $ pip install datapkg
   
   If you want it in a nice insulated virtualenv do instead:
   
   # set up virtualenv
   $ virtualenv your_virtual_env
   
   # EITHER (with easy_install)
   $ . your_virtual_env/bin/activate
   $ easy_install datapkg
   
   # OR (with pip)
   $ pip -E your_virtual_env install datapkg
   
   NB: if you want to install from source datapkg’s mercurial repository is here:
   http://knowledgeforge.net/ckan/datapkg
4. Extras (e.g. upload capabilities).
   
   If you want to use the upload capabilities you will need to install the OFS library:
   
   $ easy_install ofs
   # or
   $ pip install ofs
   
   You can also install plugins - see the project’s home page for a current list.
5. Take a look at the manual:
   
   $ datapkg man
GETTING STARTED: USERS

We recommend users get started by taking a look at the manual:

2.1 Datapkg Manual

2.2 Using datapkg from code using the API

2.2.1 Datapkg API

How to use datapkg from code (python).

2.2.2 Package ‘Specs’ (Specifications)

2.3 More information on specific features

2.3.1 Distributions

A Distribution is a serialization of the Package and optionally the data too (code, database, a book etc) to some concretely addressable form that can be ‘distributed’ (e.g. uploaded) or accessed (e.g. downloaded). For example:

- file(s) on disk
- an API at a specific url.

Note: To the extent possible, we seek to reuse existing ‘distribution’ formats rather than invent our own.

Currently, we provide 2 types of basic file distributions:

- Simple (Ini-Based) Distribution - DEFAULT
- Python Distribution

It is also easy to extend datapkg to support new distribution types. See Extending Datapkg.
Base Distribution

Simple Distribution

Python Distribution
CHAPTER THREE

GETTING STARTED: DEVELOPERS

Source git repository can be found at: http://github.com/okfn/datapkg

For developers we recommend starting with the design document:

3.1 Datapkg Design

3.1.1 Overview

The following diagram gives a conceptual overview of the datapkg system. Most, though not all, of the classes indicated already exist within datapkg.

```
DataPackage
+metadata
+resources
```

The central object in datapkg is a Package. A Package consists of:

- metadata
One important thing to emphasize is that in datapkg the payload is often not directly available (since it may be very large) but is virtual, being represented by, for example, links to the payload or a listing of the items in the payload (the manifest). In addition to the basic metadata about a package (name, version, title, description etc) we also need to record things like dependencies. Thus a Package main attributes become:

- basic
- dependencies
- manifest

Distributions are serialized Package on disk (or elsewhere) and handle all serialization and deserialization of packages.

- Other people can provide plugins (datapkg, distribution)
- These are tried in turn when loading from disk

Tools include downloaders, unpacking and the command line interface.

An Index represents a list of packages whether locally (like DbIndex) or remotely (like CkanIndex). A Repository is an Index plus storage/installation capacity.

### 3.1.2 Distributions and Metadata


There are also a set of use cases and research on other similar tools:

### 3.2 Use cases for datapkg

Use cases for DataPkg (or: reasons to use it in the first place)

These use cases are not necessarily all implemented but are a guide to what we are trying to do. The first two were the two original use cases at the start of the project (and were heavily inspired by debian).

#### 3.2.1 1. Grabbing some data from an index

The steps involved:

```bash
$ datapkg index-add file:///....
$ datapkg update
$ datapkg search "military spending"
```

* some-id Military Spending 1890-1914
* some-id-2 Military Spending 1890-1914 (normalized)

```bash
$ datapkg install some-id
...
$ datapkg plot some-id
```

#### 3.2.2 2. Get two different datasets and use them together

What data?
• Normalize data * Cross country and then convert to standard (e.g. US$, GBP)
  – Exchange rates
  – Cost of living
  – Changes across time and then do real present value
• [Plot two different data sources again each other.] * [Government expenditure in different sectors?]

Example code:

```
$ datapkg install pkg-a
$ datapkg install pkg-b
$ datapkg create merged
  # manual merge
  # e.g. PPP, GDP
$ datapkg register my-merged-package
```

### 3.2.3 Getting data v2

Revist basic discovery and usage of data from above.

1. Install datapkg
2. Search remote registry/repo for a package
3. Download package on to local disk and unpack:

   ```
   $ datapkg get [url|name] [path]
   ```

   If specifying name (using a Registry) then:
   • get metadata from registry
   • locate the distribution URL

Basic steps:
• Discover at URL: targz/zip file, version controlled repo, URL page with links
  (ask user which one)
• download the compressed distribution to temp dir (progress bar)
• unpack it to destination path

Future: maybe need to build/compile data

4. Explore package

#### Publisher user role

1. Package a csv file
2. Register the package to the remote repo.
3. Upload the package distribution to the remote repo.
3.3 Research

3.3.1 Python

Basic distribution format is zip or tar.gz. Metadata rather hacked (originally just distributed code files not metadata).

- Metadata (setup.py and/or PKG-INFO)
- Payload: files (including) data specified usually using combination of setup.py and MANIFEST/MANIFEST.in (manifest template)
- setuptools and egg-info is a bit of a mess (see below and our wrapper code) but getting sorted out (as of late 2009)

Further documentation:

1. Misc
   - A Database of Installed Python Packages: <http://www.python.org/dev/peps/pep-0262/>
   - Presentation: https://dfwpython.org/static/pycon/1-python-distutils.pdf

2. Setuptools

- Observations:
  - REALLY hard to work what is going on. E.g.
  - How does .easy-install.pth stuff work.

- Without doing custom install http://peak.telecommunity.com/DevCenter/EasyInstall#custom-installation-locations cannot get all the easy_install magic of looking stuff up by name (as this involves messing with python binary or site packages). So just install and get the directory path.

3.3.2 Debian (and R)

TODO: write-up

- Built format is tar
- metadata in ini-file + resources (files)

3.3.3 OPeNDAP (open network data access protocol)

Mostly culled from http://www.opendap.org/user/quick-html/quick_1.html

Originally developed by oceanographic institute as DODS (Distributed Oceanographic Data System) as a protocol for accessing and serving data in a manner that is transparent for remote users.

Main aspects are:

1. Metadata
   1. Dataset Descriptor Structure (DDS) – Structure of the data. “This provides a description of the “shape” of the data, using a vaguely C-like syntax. You get a dataset’s DDS by appending .dds to the URL.”
2. Data Attribute Structure (DAS) – “contains information about the data, such as units and the name of the variable”
   • “NOTE: The DAS is populated at the data provider’s discretion. Because of this, the quality of the data in it (the metadata) varies widely. The data in the Reynolds dataset used in this example are COARDS compliant. Other metadata standards you may encounter with DODS data are HDF-EOS, EPIC, FGDC, or no metadata at all.”

3. Info service: append .info to the url to get back DDS and DAS

2. Data:
   • html query: Append .html to the URL, and you get a form that directs you to add information to sample the data at a URL

3. Querying. query string: {url}?{array-name}[{restriction ...}]
   • e.g. ...sst/mnmean.nc.asc?time[0:6]

Example: http://www.cdc.noaa.gov/cgi-bin/nph-nc/Datasets/reynolds_sst/sst.mnmean.nc

3.3.4 Virtual Observatory
   • Sharing datasets in astronomy
   • Developed a data catalogue and data search and retrieval service

3.3.5 OAI-PMH
   Protocol for retrieving bibliographic catalogue data especially in relation to (research) papers.
   • Metadata: dublin-core based (XML)
   • Payload/resources: not relevant as not transmitted (metadata only service) – though can obviously point to resources.

3.4 Extending Datapkg

Datapkg has a rich plugin architecture that makes it easy to extend.

3.4.1 Extending Datapkg

Datapkg has been designed to be easily extensible. At the present time you can write your own implementations of:
   • Commands - extend datapkg command line interface with new commands
   • Indexes - add new Indexes with which datapkg can communicate
   • Distribution - add new Distribution types (either for reading or writing or both)
   • (Package) Resource downloader - add support for downloading different types of resources
   • Uploader (via OFS) - upload to different storage backends
Commands

It is easy to add your own custom commands to the set of commands available from the `datapkg` command line interface.

To provide a new command named ‘mycommand’:

1. Create a new command class inheriting from `datapkg.cli.Command`. This may be called anything you want. Assume it is called ‘MyNewCommand’ in package mynewpackage.command

2. In the `setup.py` of your new python package (containing the new command) add to the `datapkg.cli` entry poing section and entry named ‘mycommand’:

   ```
   [datapkg.cli]
   mycommand = mynewpackage.command:MyNewCommand
   ```

Command Base Class

Index

To provide a new Index for datapkg to use (e.g. in datapkg search and datapkg download commands) you must:

1. Create a new Index class inheriting from `datapkg.index.IndexBase` (see below)

2. Add an entry point for your Index class in the `[datapkg.index]` section of your `setup.py` entry_points.

   • NB: the index will be available in datapkg commands (such as search) via the entry point name. E.g. if the entry point section looks like:

   ```
   [datapkg.index]
   mynewindex = mypackage.customindex:CustomIndex
   ```

   then the can be used in datapkg commands as follows:

   ```
   $ datapkg search mynewindex:// {search-tem}
   ```

Index Base class

Distributions

To provide a new Distribution (either for reading, writing or both) for datapkg to use you must:

1. Create a new Distribution class inheriting from `datapkg.distribution.DistributionBase` (see below)

2. Add an entry point for your Index class in the `[datapkg.distribution]` section of your `setup.py` entry_points.

Distribution Base class

Resource Downloader

Uploading

datapkg utilizes the pluggable blobstore library OFS (http://bitbucket.org/okfn/of). To add a new storage backend just extend OFS and this new backend will be automatically available to datapkg.
4.1 License

All material is licensed under the MIT License:

Copyright (c) 2005-2009, Open Knowledge Foundation

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

4.2 History

4.2.1 HEAD

• Standardize on JSON-based distribution (and remove ini-based distribution)

4.2.2 V0.8 2011-02-09

• ResourceDownloader objects and plugin point (#964)
• Refactor PackageDownloader to use ResourceDownloader and support Resource filtering
• Retrieval options for package resources (#405). Support selection of resources to download (on command line or API) via glob style patterns or user interaction.
4.2.3 V0.7.1 2010-12-02

- MINOR: soften dependency on ckanclient to \(\geq 0.3\) for better compatibility with CKAN

4.2.4 V0.7 2010-10-11

- MAJOR: Support for uploading datapkgs (upload.py)
- MAJOR: Much improved and extended documentation
- MAJOR: New sqlite-based DB index giving support for a simple, central, ‘local’ index (ticket:360)
- MAJOR: Make datapkg easily extendable
  - Support for adding new Index types with plugins
  - Support for adding new Commands with command plugins
  - Support for adding new Distributions with distribution plugins
- Improved package download support (also now pluggable)
- Reimplement url download using only python std lib (removing urlgrabber requirement and simplifying installation)
- Improved spec: support for db type index + better documentation
- Better configuration management (especially internally)
- Reduce dependencies by removing dependency on PasteScript and PasteDeploy (#98)
- Various minor bugfixes and code improvements

4.2.5 V0.6.1 2010-05-04

- Bugfix release to address breaking changes in python \(\geq 2.6.5\) - see http://bugs.python.org/issue7904

4.2.6 V0.6 2010-04-28

- Fixes for unicode in package metadata (when writing to disk)
- Remove use of ast module as only in python 2.5/2.6
- Upgrade to use ckanclient 0.3
- Improved python api for usage of datapkg (load_package and load_index)

4.2.7 V0.5 2010-02-12

- Improve downloading of package resources (and remove dependency on pip)
- Improved installation with metadata written consistently (install now really, really works!)
- Register to disk works (writes metadata to disk)
- Continued improvements to CLI and documentation
4.2.8 V0.4.1 2010-01-19

- Restrict to pip<=0.6.1 (just released pip 0.6.2 has major code re-layout which causes breakages)

4.2.9 V0.4 2009-12-28

- Major refactoring to simplify and standardize CLI using 'package specs'
- **Support for searching indices from CLI, e.g.:::** datapkg search ckan:// myquery
- Simple installation onto disk
- Overhauled and improved documentation and put docs online

4.2.10 V0.3 2009-08-25

This release features major improvements over v0.2 particularly in area of CKAN integration with ‘write’ support especially improved.

- Better and more flexible system for reading and writing packages to disk
  - New Distribution object to encapsulate writing and reading to disk
  - Implementations for Python packages (PythonDistribution) and simple ini file (metadata.txt) format (Ini-BasedDistribution)
- Improved approach for configuration with config stored in a dedicated ini file and lots of minor fixes to improve CLI experience
- Create new Index object SimpleIndex
- Completely overhaul package metadata with support for many additional attributes (author, maintainer, extras etc)
- Improved, sphinx-built docs (http://knowledgeforge.net/ckan/doc/datapkg/)

4.2.11 v0.2 2009-03-18

- Completely refactor CLI making it better documented and easier to use
- Support for ‘flat’ package structure on disk
- Improve creation of on disk package (MANIFEST.in etc)
- CKAN integration fully functional
- Almost all commands now functioning and tested

4.2.12 v0.1 2008-12-15

- Manual much improved
- CLI: info and dump commands working (r331)
- Substantial improvements to interface to setuptools and easy_install (pypkgtools)
- First official released to PyPI
- Several bugs fixed
4.2.13 v0.0.5 2008-07-15

- Start on manual
- Core objects mostly working: Register, Repository, Package
- Integration with CKAN (not tested)
- Lots of tests
- Basic functions but a way to go for proper upload/download cycle

4.2.14 2007-06-20: Project Started

4.3 Notes on External Libraries

4.3.1 pip

As pip is a command line tool not a library internal docs are rather poor so we provide an overview here:

class PackageFinder(object):
    This finds packages.
    This is meant to match easy_install’s technique for looking for packages, by reading pages and looking for appropriate links

class RequirementSet(object):
    More interesting to us as has tools for downloading and unpacking ...
    def __init__(self, build_dir, src_dir, download_dir, download_cache=None,
                 upgrade=False, ignore_installed=False,
                 ignore_dependencies=False):
        src_dir, build_dir not used in unpack_*
        (used in install/install_files, create_bundle)

        def unpack_url(self, link, location, only_download=False):

Downloading: Pages, VCS Systems etc

class Link(object):
    Represents a url with a bit of extra info

    def __init__(self, url, comes_from=None):

class VcsSupport
    cache vcs support information and override urlparse in useful ways

class VersionControl
    base class with various implementations e.g. Subversion, Mercurial etc

class HTMLPage

class PageCache
 Probably not relevant to us

class InstallRequirement
    Looks like it handles actually installing something!
    ~600 lines of code
CHAPTER
FIVE

INDICES AND TABLES

• genindex
• modindex
• search