1 About the components
  1.1 Spatial Search ................................................................. 3
  1.2 Spatial Search Widget ...................................................... 4
  1.3 Dataset Extent Map ............................................................ 4
  1.4 WMS Preview .................................................................. 4
  1.5 CSW Server ....................................................................... 4
  1.6 GEMINI Harvesters .............................................................. 5
  1.7 Harvest Metadata API ......................................................... 5
  1.8 CSW Client ...................................................................... 5
  1.9Validators ........................................................................ 5
  1.10 cswninfo tool .................................................................... 6
  1.11 Validator ........................................................................ 6

2 Setup
  2.1 Install Python ................................................................... 7
  2.2 Install System Packages ......................................................... 7
  2.3 Configuration ..................................................................... 8
  2.4 Configuration - Dataset Extent Map .......................................... 8
  2.5 Configuration - CSW Server .................................................. 8
  2.6 SOLR Configuration .............................................................. 9

3 Troubleshooting ..................................................................... 11

4 Tests ...................................................................................... 13

5 Command line interface ................................................................ 15

6 Setting up PostGIS
  6.1 PostGIS Configuration .......................................................... 17
  6.2 Migrating to an existing PostGIS database .................................. 18
  6.3 Setting up a spatial table ....................................................... 18

7 Installing libxml2 .................................................................... 21

8 Licence ..................................................................................... 23
This extension contains plugins that add geospatial capabilities to CKAN. The following plugins are currently available:

- Spatial model for CKAN datasets and automatic geo-indexing (`spatial_metadata`)
- Spatial Search - Spatial search integration and API call (`spatial_query`).
- Spatial Search Widget - Map widget integrated on the search form (`spatial_query_widget`).
- Dataset Extent Map - Map widget showing a dataset extent (`dataset_extent_map`).
- WMS Preview - a Web Map Service (WMS) previewer (`wms_preview`).
- CSW Server - a basic CSW server - to server metadata from the CKAN instance (`cswserver`).
- GEMINI Harvesters - for importing INSPIRE-style metadata into CKAN (`gemini_csw_harvester`, `gemini_doc_harvester`, `gemini_waf_harvester`).
- Harvest Metadata API - a way for a user to view the harvested metadata XML, either as a raw file or styled to view in a web browser (`spatial_harvest_metadata_api`).

These libraries:

- CSW Client - a basic client for accessing a CSW server
- Validators - uses XSD / Schematron to validate geographic metadata XML. Used by the GEMINI Harvesters
- Validators for ISO19139/INSPIRE/GEMINI2 metadata. Used by the Validator.

And these command-line tools:

- `cswinfo` - a command-line tool to help making requests of any CSW server

As of October 2012, ckanext-csw and ckanext-inspire were merged into this extension.
CHAPTER ONE

ABOUT THE COMPONENTS

1.1 Spatial Search

To enable the spatial query you need to add the `spatial_query` plugin to your ini file (See Configuration). This plugin requires the `spatial_metadata` plugin.

The extension adds the following call to the CKAN search API, which returns datasets with an extent that intersects with the bounding box provided:

/api/2/search/dataset/geo?bbox={minx,miny,maxx,maxy}[&crs={srid}]

If the bounding box coordinates are not in the same projection as the one defined in the database, a CRS must be provided, in one of the following forms:

- urn:ogc:def:crs:EPSG::4326
- EPSG:4326
- 4326

As of CKAN 1.6, you can integrate your spatial query in the full CKAN search, via the web interface (see the Spatial Search Widget) or via the action API, e.g.:

POST http://localhost:5000/api/action/package_search
{
    "q": "Pollution",
    "extras": {
        "ext_bbox": "-7.535093,49.208494,3.890688,57.372349"
    }
}

1.1.1 Geo-Indexing your datasets

In order to make a dataset queryable by location, an special extra must be defined, with its key named ‘spatial’. The value must be a valid GeoJSON geometry, for example:

{"type": "Polygon", "coordinates": [[[2.05827, 49.8625], [2.05827, 55.7447], [-6.41736, 55.7447], [-6.41736, 49.8625], [2.05827, 49.8625]]]

or:

{"type": "Point", "coordinates": [-3.145,53.078] }

Every time a dataset is created, updated or deleted, the extension will synchronize the information stored in the extra with the geometry table.
1.2 Spatial Search Widget

Note: this plugin requires CKAN 1.6 or higher.

To enable the search map widget you need to add the `spatial_query_widget` plugin to your ini file (See Configuration). You also need to load both the `spatial_metadata` and the `spatial_query` plugins.

When the plugin is enabled, a map widget will be shown in the dataset search form, where users can refine their searches drawing an area of interest.

1.3 Dataset Extent Map

To enable the dataset map you need to add the `dataset_extent_map` plugin to your ini file (See Configuration). You need to load the `spatial_metadata` plugin also.

When the plugin is enabled, if datasets contain a ‘spatial’ extra like the one described in the previous section, a map will be shown on the dataset details page.

1.4 WMS Preview

To enable the WMS previewer you need to add the `wms_preview` plugin to your ini file (See Configuration).

Please note that this is an experimental plugin and may be unstable.

When the plugin is enabled, if datasets contain a resource that has ‘WMS’ format, a ‘View available WMS layers’ link will be displayed on the dataset details page. It forwards to a simple map viewer that will attempt to load the remote service layers, based on the GetCapabilities response.

1.5 CSW Server

CSW (Catalogue Service for the Web) is an OGC standard for a web interface that allows you to access metadata (which are records that describe data or services)

The currently supported methods with this CSW Server are:

- GetCapabilities
- GetRecords
- GetRecordById

ckanext-csw provides the CSW service at /csw.

For example you can ask the capabilities of the CSW server installed into CKAN running on 127.0.0.1:5000 like this:

curl ‘http://127.0.0.1:5000/csw?request=GetCapabilities&service=CSW’

The standard CSW response is in XML format.
1.6 GEMINI Harvesters

These harvesters were designed to harvest metadata records in the GEMINI2 format, which is an XML spatial metadata format very similar to ISO19139. This was developed for the UK Location Programme and GEMINI2, but it would be simple to adapt them for other INSPIRE or ISO19139-based metadata.

The harvesters get the metadata from these types of server:

- GeminiCswHarvester - CSW server
- GeminiWafHarvester - WAF file server - An index page with links to GEMINI resources
- GeminiDocHarvester - HTTP file server - An individual GEMINI resource

The GEMINI-specific parts of the code are restricted to the fields imported into CKAN, so it would be relatively simple to generalise these to other INSPIRE profiles.

Each contains code to do the three stages of harvesting:

- gather_stage - Submits a request to Harvest Sources and assembles a list of all the metadata URLs (since each CSW record can recursively refer to more records?). Some processing of the XML or validation may occur.
- fetch_stage - Fetches all the Gemini metadata
- import_stage - Validates all the Gemini, converts it to a CKAN Package and saves it in CKAN

You must specify which validators to use in the configuration of ckan.spatial.validator.profiles - see below.

1.7 Harvest Metadata API

Enabled with the ckan.plugins = spatial_harvest_metadata_api (previous known as inspire_api)

To view the harvest objects (containing the harvested metadata) in the web interface, these controller locations are added:

/api/2/rest/harvestobject/<id>/xml
/api/2/rest/harvestobject/<id>/html

1.8 CSW Client

CswService is a client for python software (such as the CSW Harvester in ckanext-inspire) to conveniently access a CSW server, using the same three methods as the CSW Server supports. It is a wrapper around OWSLib’s tool, dealing with the details of the calls and responses to make it very convenient to use, whereas OWSLib on its own is more complicated.

1.9 Validators

This library can validate metadata records. It currently supports ISO19139 / INSPIRE / GEMINI2 formats, validating them with XSD and Schematron schemas. It is easily extensible.

To specify which validators to use during harvesting, specify their names in CKAN config. e.g.:
ckan.spatial.validator.profiles = iso19139,gemini2,constraints

1.10 cswinfo tool

When ckanext-csw is installed, it provides a command-line tool cswinfo, for making queries on CSW servers and returns the info in nicely formatted JSON. This may be more convenient to type than using, for example, curl.

Currently available queries are:

- getcapabilities
- getidentifiers
- getrecords
- getrecordbyid

For details, type:

cswinfo csw -h

There are options for querying by only certain types, keywords and typenames as well as configuring the ElementSetName.

The equivalent example to the one above for asking the capabilities is:

```bash
$ cswinfo csw getcapabilities http://127.0.0.1:5000/csw
```

OWSLib is the library used to actually perform the queries.

1.11 Validator

This python library uses Schematron and other schemas to validate the XML.

Here is a simple example of using the Validator library:

```python
from ckanext.csw.validation import Validator
xml = etree.fromstring(gemini_string)
validator = Validator(profiles=('iso19139', 'gemini2', 'constraints'))
valid, messages = validator.isvalid(xml)
if not valid:
    print "Validation error: " + messages[0] + ':
    ' + '
    '.join(messages[1:])
```

In DGU, the Validator is integrated here: https://github.com/okfn/ckanext-inspire/blob/master/ckanext/inspire/harvesters.py#L88

NOTE: The ISO19139 XSD Validator requires system library libxml2 v2.9 (released Sept 2012). If you intend to use this validator then see the section below about installing libxml2.
CHAPTER TWO

SETUP

2.1 Install Python

1. Install this extension into your python environment (where CKAN is also installed).

   *Note:* Depending on the CKAN core version you are targeting you will need to use a different branch from the extension.

   For a production site, use the *stable* branch, unless there is a specific branch that targets the CKAN core version that you are using.

   To target the latest CKAN core release:
   
   (pyenv) $ pip install -e git+https://github.com/okfn/ckanext-spatial.git@stable#egg=ckanext-spatial

   To target an old release (if a release branch exists, otherwise use *stable)*:
   
   (pyenv) $ pip install -e git+https://github.com/okfn/ckanext-spatial.git@release-v1.8#egg=ckanext-spatial

   To target CKAN *master*, use the extension *master* branch (ie no branch defined):
   
   (pyenv) $ pip install -e git+https://github.com/okfn/ckanext-spatial.git#egg=ckanext-spatial

   *cswserver* requires that *ckanext-harvest* is also installed (and enabled) - see [https://github.com/okfn/ckanext-harvest](https://github.com/okfn/ckanext-harvest)

2. Install the rest of python modules required by the extension:

   (pyenv) $ pip install -r pip-requirements.txt

2.2 Install System Packages

There are also some system packages that are required:

- PostGIS and must be installed and the database needs spatial features enabling to be able to use Spatial Search. See the "Setting up PostGIS" section for details.

- Shapely requires libgeos to be installed. If you installed PostGIS on the same machine you have already got it, but if PostGIS is located on another server you will need to install GEOS on it:

  
  sudo apt-get install libgeos-c1

- The Validator for ISO19139 requires the install of a particular version of libxml2 - see “Installing libxml2” for full details.
2.3 Configuration

Once PostGIS is installed and configured in your database (see the “Setting up PostGIS” section for details), you need to create some DB tables for the spatial search, by running the following command (with your python env activated):

```bash
(pyenv) $ paster --plugin=ckanext-spatial spatial initdb [srid] --config=mysite.ini
```

You can define the SRID of the geometry column. Default is 4326. If you are not familiar with projections, we recommend to use the default value.

Check the Troubleshooting section if you get errors at this stage.

Each plugin can be enabled by adding its name to the `ckan.plugins` in the CKAN ini file. For example:

```
ckan.plugins = spatial_metadata spatial_query spatial_query_widget dataset_extent_map wms_preview
```

**Note:** Plugins `spatial_query`, `spatial_query_widget` and `dataset_extent_map` depend on the `spatial_metadata` plugin also being enabled.

When enabling the spatial metadata, you can define the projection in which extents are stored in the database with the following option. Use the EPSG code as an integer (e.g 4326, 4258, 27700, etc). It defaults to 4326:

```ini
ckan.spatial.srid = 4326
```

2.4 Configuration - Dataset Extent Map

If you want to define a default map extent for the different map widgets, (e.g. if you are running a national instance of CKAN) you can do so adding this configuration option:

```ini
ckan.spatial.default_map_extent=<minx>,<miny>,<maxx>,<maxy>
```

Coordinates must be in latitude/longitude, e.g.:

```ini
ckan.spatial.default_map_extent=-6.88,49.74,0.50,59.2
```

The Dataset Extent Map displays only on certain routes. By default it is just the ‘Package’ controller, ‘read’ method. To display it on other routes you can specify it in a space separated list like this:

```ini
ckan.spatial.dataset_extent_map.routes = package/read ckanext.dgu.controllers.package:PackageController/read
```

The Dataset Extent Map provides two different map types. It defaults to ‘osm’ but if you have a license and apikey for ‘os’ then you can use that map type using this configuration:

```ini
ckan.spatial.dataset_extent_map.map_type = os
```

The Dataset Extent Map will be inserted by default at the end of the dataset page. This can be changed by supplying an alternative element_id to the default:

```ini
ckan.spatial.dataset_extent_map.element_id = dataset
```

2.5 Configuration - CSW Server

Configure the CSW Server with the following keys in your CKAN config file (default values are shown):

```ini
```
ckanext-spatial Documentation, Release 0.1

cswservice.title = Untitled Service - set cswservice.title in config
cswservice.abstract = Unspecified service description - set cswservice.abstract in config
cswservice.keywords =
cswservice.keyword_type = theme
cswservice.provider_name = Unnamed provider - set cswservice.provider_name in config
cswservice.contact_name = No contact - set cswservice.contact_name in config
cswservice.contact_position =
cswservice.contact_voice =
cswservice.contact_fax =
cswservice.contact_address =
cswservice.contact_city =
cswservice.contact_region =
cswservice.contact_pcode =
cswservice.contact_country =
cswservice.contact_email =
cswservice.contact_hours =
cswservice.contact_instructions =
cswservice.contact_role =
cswservice.rndlog_threshold = 0.01
cswservice.log_xml_length = 1000

cswservice.rndlog_threshold is the percentage of interactions to store in the log file.

2.6 SOLR Configuration

If using Spatial Query functionality then there is an additional SOLR/Lucene setting that should be used to set the limit on number of datasets searchable with a spatial value.

The setting is `maxBooleanClauses` in the `solrconfig.xml` and the value is the number of datasets spatially searchable. The default is 1024 and this could be increased to say 16384. For a SOLR single core this will probably be at `/etc/solr/conf/solrconfig.xml`. For a multiple core set-up, there will me several solrconfig.xml files a couple of levels below `/etc/solr`. For that case, **ALL** of the cores’ `solrconfig.xml` should have this setting at the new value.

Example:

```xml
<maxBooleanClauses>16384</maxBooleanClauses>
```

This setting is needed because PostGIS spatial query results are fed into SOLR using a Boolean expression, and the parser for that has a limit. So if your spatial area contains more than the limit (of which the default is 1024) then you will get this error:

```
Dataset search error: ('SOLR returned an error running query...
```

and in the SOLR logs you see:

```
too many boolean clauses
...
Caused by: org.apache.lucene.search.BooleanQuery$TooManyClauses: maxClauseCount is set to 1024
```
TROUBLESHOOTING

Here are some common problems you may find when installing or using the extension:

• When initializing the spatial tables:

  ```sql
  LINE 1: SELECT AddGeometryColumn('package_extent','the_geom', E'4326...'
  ^
  HINT: No function matches the given name and argument types. You might need to add explicit type
  "SELECT AddGeometryColumn('package_extent','the_geom', %s, 'GEOMETRY', 2)" ('4326',)
  
  PostGIS was not installed correctly. Please check the “Setting up PostGIS” section.
  ```

  ```python
  sqlalchemy.exc.ProgrammingError: (ProgrammingError) permission denied for relation spatial_ref_sys
  
  The user accessing the ckan database needs to be owner (or have permissions) of the geometry_columns and
  spatial_ref_sys tables.
  ```

• When performing a spatial query:

  ```python
  InvalidRequestError: SQL expression, column, or mapped entity expected - got `<class 'ckanext.spatial.model.PackageExtent'>`
  
  The spatial model has not been loaded. You probably forgot to add the spatial_metadata plugin to your ini
  configuration file.
  ```

  ```python
  InternalError: (InternalError) Operation on two geometries with different SRIDs
  
  The spatial reference system of the database geometry column and the one used by CKAN differ. Remember, if
  you are using a different spatial reference system from the default one (WGS 84 lat/lon, EPSG:4326), you must
  define it in the configuration file as follows:
  ```

  ```python
  ckan.spatial.srid = 4258
  ```
All of the tests need access to the spatial model in Postgres, so to run the tests, specify `test-core.ini`:

```
(pyenv) $ nosetests --ckan --with-pylons=test-core.ini -l ckanext ckanext/spatial/tests
```

In some places in this extension, ALL exceptions get caught and reported as errors. Since these could be basic coding errors, to aid debugging these during development, you can request exceptions are reraised by setting the DEBUG environment variable:

```
export DEBUG=1
```
COMMAND LINE INTERFACE

The following operations can be run from the command line using the `paster spatial` command:

`initdb [srId]`
- Creates the necessary tables. You must have PostGIS installed and configured in the database.
  You can provide the SRID of the geometry column. Default is 4326.

`extents`
- Creates or updates the extent geometry column for datasets with an extent defined in the ‘spatial’ extra.

The commands should be run from the ckanext-spatial directory and expect a development.ini file to be present. Most of the time you will specify the config explicitly though:

`paster spatial extents --config=./ckan/development.ini`
6.1 PostGIS Configuration

- Install PostGIS:
  ```
  sudo apt-get install postgresql-8.4-postgis
  (or postgresql-9.1-postgis, depending on your postgres version)
  ```
- Create a new PostgreSQL database:
  ```
  sudo -u postgres createdb [database]
  ```
  (If you just want to spatially enable an existing database, you can ignore this point, but it’s a good idea to create a template to make easier to create new databases)
- Many of the PostGIS functions are written in the PL/pgSQL language, so we need to enable it in our database:
  ```
  sudo -u postgres createlang plpgsql [database]
  ```
- Run the following commands. The first one will create the necessary tables and functions in the database, and the second will populate the spatial reference table:
  ```
  sudo -u postgres psql -d [database] -f /usr/share/postgresql/8.4/contrib/postgis-1.5/postgis.sql
  sudo -u postgres psql -d [database] -f /usr/share/postgresql/8.4/contrib/postgis-1.5/spatial_ref_sys.sql
  ```
  **Note:** depending on your distribution and PostGIS version, the scripts may be located on a slightly different location, e.g.:
  ```
  /usr/share/postgresql/8.4/contrib/postgis.sql
  ```
- Execute the following command to see if PostGIS was properly installed:
  ```
  sudo -u postgres psql -d [database] -c "SELECT postgis_full_version()"
  ```
  You should get something like:
  ```
  postgis_full_version
  POSTGIS="1.5.2" GEOS="3.2.2-CAPI-1.6.2" PROJ="Rel. 4.7.1, 23 September 2009" LIBXML="2.7.7" USE_ (1 row)
  ```
  Also, if you log into the database, you should see two tables, `geometry_columns` and `spatial_ref_sys` (and probably a view called `geography_columns`).
Note: This commands will create the two tables owned by the postgres user. You probably should make owner the user that will access the database from ckan:

```
ALTER TABLE spatial_ref_sys OWNER TO [your_user];
ALTER TABLE geometry_columns OWNER TO [your_user];
```

More information on PostGIS installation can be found here:

http://postgis.refractions.net/docs/ch02.html#PGInstall

### 6.2 Migrating to an existing PostGIS database

If you are loading a database dump to an existing PostGIS database, you may find errors like

```
ERROR: type "spheroid" already exists
```

This means that the PostGIS functions are installed, but you may need to create the necessary tables anyway. You can force psql to ignore these errors and continue the transaction with the ON_ERROR_ROLLBACK=on:

```
sudo -u postgres psql -d [database] -f /usr/share/postgresql/8.4/contrib/postgis-1.5/postgis.sql -v ON_ERROR_ROLLBACK=on
```

You will still need to populate the spatial_ref_sys table and change the tables permissions. Refer to the previous section for details on how to do it.

### 6.3 Setting up a spatial table

**Note:** If you run the `initdb` command, the table was already created for you. This section just describes what’s going on for those who want to know more.

To be able to store geometries and perform spatial operations, PostGIS needs to work with geometry fields. Geometry fields should always be added via the `AddGeometryColumn` function:

```
CREATE TABLE package_extent(
    package_id text PRIMARY KEY
);
ALTER TABLE package_extent OWNER TO [your_user];
SELECT AddGeometryColumn('package_extent','the_geom', 4326, 'GEOMETRY', 2);
```

This will add a geometry column in the `package_extent` table called `the_geom`, with the spatial reference system EPSG:4326. The stored geometries will be polygons, with 2 dimensions (The actual table on CKAN uses the GEOMETRY type to support multiple geometry types).

Have a look a the table definition, and see how PostGIS has created three constraints to ensure that the geometries follow the parameters defined in the geometry column creation:

```
# \d package_extent

Table "public.package_extent"
Column | Type | Modifiers
---------+----------+-----------
package_id | text | not null
the_geom | geometry |
Indexes:  "package_extent_pkey" PRIMARY KEY, btree (package_id)
```
Check constraints:
   "enforce_dims_the_geom" CHECK (st_ndims(the_geom) = 2)
   "enforce_srid_the_geom" CHECK (st_srid(the_geom) = 4326)
INSTALLING LIBXML2

Version 2.9 is required for the ISO19139 XSD validation.

With CKAN you would probably have installed an older version from your distribution. (e.g. with `sudo apt-get install libxml2-dev`). You need to find the SO files for the old version:

$ find /usr -name "libxml2.so"

For example, it may show it here: `/usr/lib/x86_64-linux-gnu/libxml2.so`. The directory of the SO file is used as a parameter to the `configure` next on.

Download the libxml2 source:

$ cd ~
$ wget ftp://xmlsoft.org/libxml2/libxml2-2.9.0.tar.gz

Unzip it:

$ tar zxvf libxml2-2.9.0.tar.gz
$ cd libxml2-2.9.0/

Configure with the SO directory you found before:

$ ./configure --libdir=/usr/lib/x86_64-linux-gnu

Now make it and install it:

$ make
$ sudo make install

Now check the install by running xmllint:

$ xmllint --version
xmllint: using libxml version 20900
compiled with: Threads Tree Output Push Reader Patterns Writer SAXv1 FTP HTTP DTDValid HTML Legacy
This code falls under different copyrights, depending on when it was contributed and by whom:: © (c) Copyright 2011-2012 Open Knowledge Foundation * Crown Copyright * XML/XSD files: copyright of their respective owners, held in the files themselves.

All of this code is licensed for reuse under the Open Government Licence

http://www.nationalarchives.gov.uk/doc/open-government-licence/