SANSA Catalogue Documentation

Tim Sutton, 2008

Contents

1 Introduction

1.1 The CSIR

This project has been carried out for the CSIR Satellite Applications Center, near Johannesburg, South Africa. The CSIR is the 'Council for Science and Industrial Research' - it is the main national science foundation of the country. They are a big organisation with many divisions of which SAC (Satellite Applications Center) is one.

1.2 SAC

SAC is a satellite ground station. This means they have a big campus with many antennas and collect information from satellites as they pass over our sky window they also do satellite tasking (telling satellites where to go and what to do) and satellite / space craft telemetry (tracking space vehicle orbit information etc).

SAC has two divisions:

1. Telemetry command and control where they do tracking, tasking etc.
2. EO (Earth Observation) where the focus is more software based to do remote sensing and generate products from imagery downloaded from satellites

SAC-EO is the client for this project.

1.3 SANSA Takover

South Africa is busy creating its own space agency - SANSA (South AfricAN National Space Agency). SANSA will aggregate space technology from various gov, parastatal, non-gov organisations to form a new organisation funded by the state. SAC-EO is scheduled to become part of SANSA as of 1 April 2011 and wil become SANSA-EO.

1.4 The project

SAC-EO has been building for the last 3 or 4 years an integrated system before this project (of which we form a small part), the processing of imagery was done manually and ad-hoc which is not very efficient and prone to difficulty if an expert leaves.

Thus they have started to build an integrated system called SAEOS (pronounced 'sigh-os'). The purpose of SAEOS is to create an automated processing environment through all the steps of the EO product workflow i.e.:

- satellite tasking ('please programme spot5 to take an image at footprint foo on date X')
- image processing (level 1a through 3a/b)
- image analysis (level 4)

- image ordering ('can I please get a copy of that SPOT image you took on dec 4 2008 of this area')
- product packaging ('bundle up the stuff that was ordered using a DVD robot, placing on an ftp site, writing to an external HD etc')

To achieve this goal they have a number of software components.

The first components are the 'terminal software'. Terminal software are provided by satellite operators such as SPOT5 (I will use SPOT as an example a lot as its the pilot sensor for their project, eventually to incorporate many more sensors) The terminal software is typically a linux box with the operators own proprietary software on top that lets the operators do the tasking of satellites (to collect an image at a given place and time) and also to extract archived images from their tape library

The second component is 'SARMES'. SARMES is a collection of EASI scripts / routines. EASI is a programming language that runs on top of PCI / Geomatica a proprietary GIS tool that runs on windows and linux. SAC are busy porting SARMES to SARMES II which has the same functionality but uses python language bindings of PCI/Geomatica instead of EASI script. SARMES has all the logic to do things like:

- take a raw image and convert it to a common GIS format e.g. pix, gtiff etc.
- collect GCP's automatically using a reference image
- orthorectify an image using a dem, gcps and other reference data
- reproject the image into different coord systems (typically UTM 33S - UTM 36S in our area but others may apply too)
- perform atmospheric correction to remove effects of the stratosphere interference between lens and ground target
- perform sensor specific correction to e.g. remove effects of lens distortion on a specific camera (using published sensor models)
- perform mosaicking of images to create one big seamless colour corrected dataset
- perform pan sharpening (make a colour image higher resolution by merging it with a pan-chromatic / grey scale band)
- chop up images in various tile schemes (e.g. degree squares, quarter degree sqauares etc)

These jobs are run by manual process - creating config files, placing input files in a specific dir heirachy etc.

The third component is DIMS. DIMS is a software system running on top of linux written in java, corba, and using oracle or postgresql as a backend (at SAC they are using PostgreSQL). DIMS is proprietary software written by a german company called

WERUM. The same software is used by the German Space Agency and others. DIMS provides automated tool chain processing. Basically you set up work flows and run them using an 'operating tool'. Although DIMS uses postgresql, there is no third party access to that db and the whole system should be considered a black box except for a few specific entry and exit points.

DIMS is being extended and customised for SAC-EO including modifications so it wll provide ogc interfaces. Before this had their own catalogue implementation and ordering system using very old standards or proprietary interfaces. So DIMS can process EO data and it builds up a catalogue of products that it has processed or 'knows about' - in its own silo. This catalogue is / will be accessible via CSW and for processing of ordering they are implementing the OGC

The OS4EO (ordering service for earth observation) is an ogc standard. The OS4EO standard is pretty simple and familiar. In essence it allows you to:

- get capabilities
- get quote
- place order

In DIMS it is implemented using SOAP rather than a RESTful service.

Along the process of creating the SAEOS project, SAC-EO have also been investing in high end hardware - particularly storage. They have a petabyte capable heirachical storage system that in short works as follows:

- data is written to local hard drives
- after a certain period of inactivity moved down to slower sata drives (nearline storage)
- and after that its migrated down onto a tape library

The tape library (offline storage) is treated as part of the file system. It has a robot arm that loads tapes automatically. When you browse the file system, it appears that all data is local since all inodes are present in online storage. When you try to read a file that is offline, the robot fetches if from tape and puts it online - typically in under a minute, though that depends on system load.

DIMS is integrated with this file system (this file system is SGI's HFS - Heirachical File System). HFS is also proprietary software running on top of Linux. One of the things DIMS will be doing is de-archiving from old manually loaded tapes and moving them into HFS. De-archiving historically collected raw satellite imagery that is. When DIMS is finished going through that there will be hundreds of thousands (probably millions) of raw images stored in HFS and accessible via DIMS.

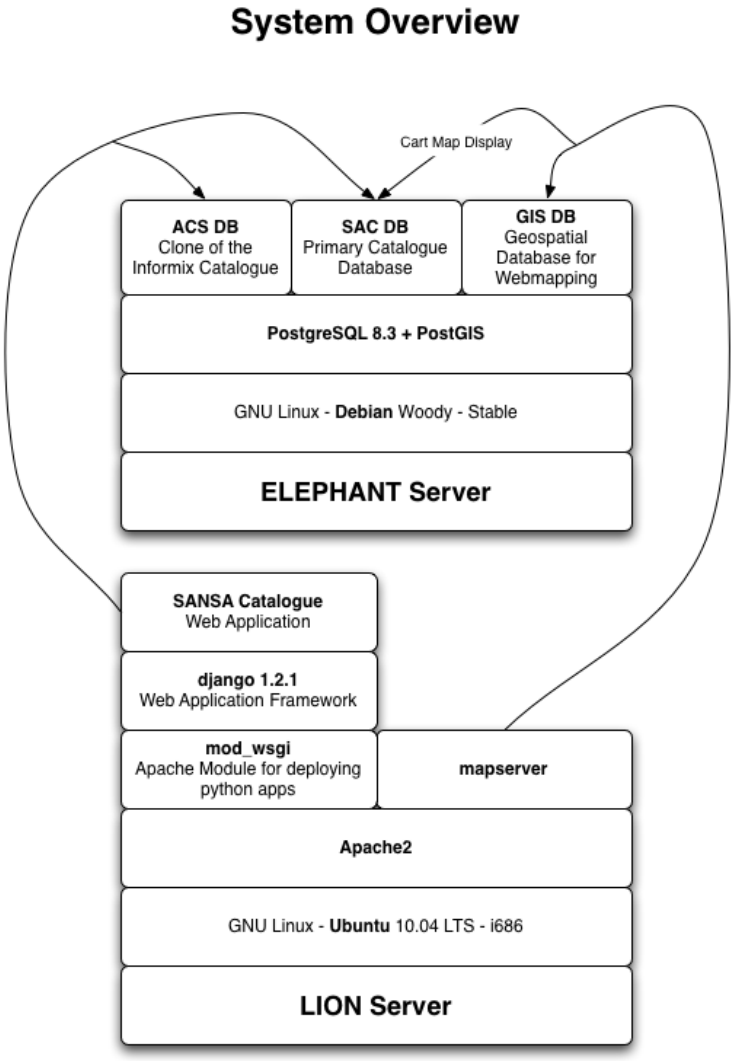
Since DIMS integrates with SARMES so you can do things like:

"Pull out that landsat 5 image from 2002, orthorectify it, correcto for atmospheric interference and lens distortions, reproject it to UTM 35S and clip it to this bounding box, then place the product on a dvd and write this label on it"

Thats the goal of the system - end to end automation with minimal operator intervention.

1.5 The Online Catalogue

Along side of these other packages, Linfiniti has been building a new web catalogue for SAC-EO. The catalogue is django + postgresql + all the other great FOSS tools we can use together to make a rich, interactive site.



Work on this started before any dims software was installed on site and the first major task was to migrate data and thumbnailss from a legacy, proprietary informix catalogue and get it into postgresql.

We also went through a few refactorings since the first version was almost a direct clone of the data model from the legacy informix system. Our current version uses the

concept of a 'generic product' for the data model.

XXXXXXXXX Insert image here XXXXXXXXXXXXXXX

On the left / center part of the diagram you will see an entity name 'Generic product'. This is a generic representation of any product (e.g. optical, radar, atmospheric, derived (like landcover map) and in the future we want to tune this to cater for vector data too.

There are five inherited models:

XXXXXXXXXXX Todo explain inherited models XXXXXXXXXXXXXXXXx

There are a bunch of small tables that provide foreign key dictionaries for the terms described in the models including:

- tables relating to ordering and tasking
- search and search record models are used to store user searches
- temporary tables used during product imports

The Online catalogue has the capability to deliver some products directly if they are held on local storage and also some basic capabilities for visitors to submit tasking requests.

Every time a search is made, its remembered, and the records the user is shown may be added to a cart and then assigned to an order To get started, first add an entry like this to your ssh config file in ~/.ssh/config:

```
Host linfiniti2
  Port 8697
  HostName 188.40.123.80
  FallBackToRsh no
```

1.6 Checkout Sources

Then setup a working dir and check out the sources (you can adapt dirs / paths as needed, using these paths will keep you consistent with all setup notes but its not required).

```
cd /home
sudo mkdir web
sudo chown -R <username>.<username> web
cd web
mkdir sac
git clone git@linfiniti2:sac_catalogue.git sac_catalogue
```

Then follow the instructions in README, skipping sections on informix, building gdal from source and source code checkout (you already checked it out if you have the readme :-)

1.7 Load a database dump

A recent database dump can be obtained from:

```
http://196.35.94.243/sac_postgis_01February2011.dmp
```

2 Working with Git

Each devloper works on a remote branch, others can track a specific branch locally and try out implemented features. After approving implementation, branch is merged with HEAD. (possibly closed/removed from tree)

This commands are based on http://www.eecs.harvard.edu/~cduan/technical/git/

2.1 Getting a list of branches

For local branches do:

```
git branch -v
```

For remote branches do:

```
git branch -r -v
```

2.2 To create remote branch

For current versions of git (at least git 1.7 or better). Say we want to create a new branch called 'docs-branch':

```
git branch docs-branch
git push --set-upstream origin docs-branch
git checkout docs-branch
```

2.3 Working with a remote branch

To be able to work with a remote branch locally (if it already exists remotely), we must create local branch and setup tracking of remote branch.

```
git pull #your local repo must be up to date first
git branch --track new-branch origin/new-branch
git checkout new-branch
```

Now you can go on to do your work in that branch.

To pull changes from remote repo do:

```
git pull origin
```

2.4 Deleting branches

Once you are done with a branch, you can delete it. For a local branch do:

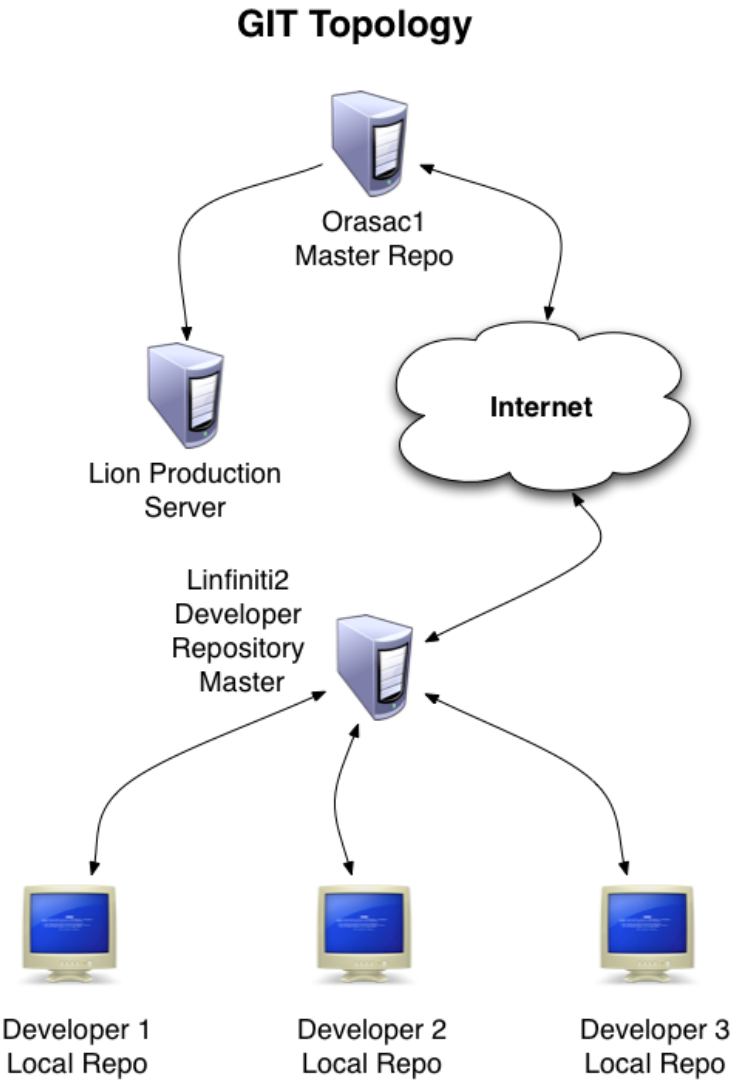
```
git branch -d new-branch
```

To delete a remote branch do (after first deleting it locally):

```
git push origin :new-branch
```

2.5 Distributed Git Repository Topology

The repositories are arranged like this:



The orasac master repo must pull from the linfiniti2 server at regular (e.g. weekly) intervals using a command like this:

```
cd /opt/git/sac_catalogue
git pull git@linfiniti2:sac_catalogue.git
```

If changes have happened on the SAC side and committed to the repository on orasac1, those changes should be pushed over to the catalogue on linfiniti2 so that the two repos are in sync:

```
cd /opt/git/sac_catalogue
git push git@linfiniti2:sac_catalogue.git
```

Note that orasac1 also has an entry in /home/timlinux/.ssh/config like this:

```
Host linfiniti2
  HostName 188.40.123.80
  User timlinux
  Port 8697
```

The lion live and test instances are cloned from the orasac1 repo like this:

```
git clone timlinux@orasac1:/opt/git/sac_catalogue sac_live
git clone timlinux@orasac1:/opt/git/sac_catalogue sac_test
```

The instance on linfiniti2 gitosis was cloned in the same way into /opt/git/repositories/sac_catalogue.

For the Tim / Drazen / Alessandro clones, the clone was carried out as described in the first section of this doc.

2.6 Tracking branches from linfiniti with a master checkout from orasac

In this scenario, we want to be tracking master from orasac1 but occationally pulling down branches from linfiniti2 to test them under lion:/opt/sac_catalogue/sac_test. Make sure you have a linfiniti2 entry in your ~/.ssh/config as described further up in this document.

```
git remote add linfiniti2 git@linfiniti2:sac_catalogue.gi
git fetch linfiniti2
```

You should see something like the output below showing you that the branches from the secondary remote repository:

```
The authenticity of host '[188.40.123.80]:8697 ([188.40.123.80]:8697)' can't be established.
RSA key fingerprint is cd:86:2b:8c:45:61:ae:15:13:45:95:25:8e:9a:6f:c4.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[188.40.123.80]:8697' (RSA) to the list of known hosts.
 _ _        __ _        _ _   _
| (_)_ __  / _(_)_ __ (_) |_(_)
| | | '_ \| |_| | '_ \| | __| |
| | | | | |  _| | | | | | |_| |
|_|_|_| |_|_| |_|_| |_|_|\__|_|

 -- Authorized Access Only --
Enter passphrase for key '/home/timlinux/.ssh/id_dsa':
remote: Counting objects: 201, done.
remote: Compressing objects: 100% (150/150), done.
remote: Total 150 (delta 103), reused 0 (delta 0)
Receiving objects: 100% (150/150), 1.10 MiB | 47 KiB/s, done.
Resolving deltas: 100% (103/103), completed with 28 local objects.
From linfiniti2:sac_catalogue
 * [new branch]      ale        -> linfiniti2/ale
 * [new branch]      ale_test   -> linfiniti2/ale_test
 * [new branch]      map_resize -> linfiniti2/map_resize
 * [new branch]      master     -> linfiniti2/master
 * [new branch]      tim-model-refactor-off-ale -> linfiniti2/tim-model-refactor-off-ale
‘
```

Now we are ready to check out the branch from there e.g.:

```
git branch map_resize linfiniti2/map_resize
git pull #not sure if needed
git checkout map_resize
sudo /etc/init.d/apache2 reload
```

When you want to get back to the original again do:

```
git checkout origin/master
```

2.7 Tracking Linfiniti in your local repo and pushing changes to orasac1

In this scenario, we want to have our master repo on the linfiniti development server, and then periodically push changes over to orasac1 production repo. Our checkout is on a third, deskop computer. So we do:

```
git clone git@linfiniti2:sac_catalogue.git sac_catalogue
```

That gives us a local repo whose remote master is on linfiniti. Now we add a new remote (you can have multiple remote repos and sync between them):

```
git remote add orasac1 timlinux@orasac1:/opt/git/sac_catalogue
git pull
```

Ok now our local repo is 'aware' of the remote repo on orasac1. So lets make a branch that tracks master on orasac1:

```
git branch --track orasac1-master orasac1/master
git checkout orasac1-master
```

Now it is simple to pull changes down from linfiniti and push them over to orasac1:

```
git merge master
git push
```

Since the branch is tracking orasac1/master they will automatically get pushed there.

3 System logic and rules

3.1 Computation of geometric_accuracy_mean

The geometric accuracy of a product is calculated as the mean of its geometric_resolution_x, geometric_resolution_y.

The values for geometric_resolution_x, geometric_resolution_y will vary per sensor and per mode. According to the following table:

(SAC to provide required detail here).

——— TABLE PLACEHOLDER ————

3.2 Sensor viewing angle

The sensor viewing angle

4 Updates and imports of products

vim /mnt/cataloguestorage/thumbnail_processing/thumb_blobs/lastblob.txt

Set desired blob no in above file.

python manage.py runscript –pythonpath=scripts -v 2 acs_importer

5 Installation Guide

5.1 Prepare your system

You need to be running Django >= 1.2.1 for the catalogue to work. Ubuntu Lucid and Debian Lenny ship with older versions so do a manual build. We walk through this setup using the python virtual environment system.

5.1.1 Create working dir

```
cd /opt
mkdir sac
cd sac
```

5.1.2 Setup python virtual environment

We install Python in a virtual environment on Ubuntu and Debian, to be able to install Django 1.2 separate from the "System Python" and avoid conflicts.

If you do not have the Python virtualenv software, get it with:

```
sudo apt-get install python-virtualenv
```

Now, start the Python virtual environment setup. We install Python in the "python" subfolder of the project directory and then activate the virtual environment.

```
virtualenv --no-site-packages python
source python/bin/activate
```

5.1.3 Install some development dependencies

```
sudo apt-get install libpq-dev libpq4 libpqxx-dev
```

Install easy_install so that we can use pip thereafter:

```
easy_install pip
```

5.1.4 Informix DB Support

This is only needed on machines that will be doing updates from the legacy acs system.

You need to have the informix client sdk installed on the machine first.

Then make sure the virtual environment is active:

```
source ../python/bin/activate
```

Then extract the python informix client to tmp and install it into your venv.

```
cd /tmp/
tar xfz /home/timlinux/Informix/InformixDBPython-2.5.tar.gz
cd InformixDB-2.5/
python setup.py build_ext
python setup.py install
```

5.1.5 GDAL Python Bindings

The gdal python bindings (which are installed using the REQUIREMENTS file in the section that follows) will not compile without swq.h header. On my production servers where I am using a hand-built gdal with ecw support, I coped the aforementioned header into /usr/local/include. The header file is available here:

```
http://svn.osgeo.org/gdal/branches/1.7/gdal/ogr/swq.h
```

5.1.6 Install django and required django apps

To install django, django authentication etc into our virtual environment do:

```
pip install -r sac_catalogue/REQUIREMENTS.txt
```

Then make sure the appropriate settings from djangodblog in settings.py.templ are deployed in your production settings.py

The full list of packages installed using the REQUIREMENTS file is:

```
#Note 1.2.4 has broken routers / multidb support
django==1.2.1
django-registration
sorl-thumbnail
django-db-log
django-debug-toolbar
django-extensions
psycopg2
pil
gdata
GDAL
django-dag
html5lib==0.90
reportlab==2.5
pisa==3.0.33
pygooglechart==0.3.0
lxml
pysqlite
```

5.1.7 Further info on django registration

You may also want to read this:

http://devdoodles.wordpress.com/2009/02/16/user-authentication-with-django-registration/

if you want more info on how the registration stuff works.

Note: that you need to log in to the admin area of the site and change the domain name in the sites table from something other than 'example.com', otherwise the registration middleware will send the reminder with an incorrect url.

5.2 Source code Check out

Check out this folder using

```
svn co https://196.35.94.196/svn/trunk/sac_catalogue
cd sac_catalogue
```

Copy settings.py.template to settings.py and then modify settings.py as needed (probably you just need to set the eth adapter and db connection settings).

5.3 Database setup

Create the database using:

```
createlang plpgsql template1
psql template1 < /usr/share/postgresql-8.3-postgis/lwpostgis.sql
psql template1 < /usr/share/postgresql-8.3-postgis/spatial_ref_sys.sql
createdb sac
createdb acs
```

5.3.1 For an empty database:

Sync the model to the db (dont do this is you plan to restore an existing db as explained in the next section):

```
python manage.py syncdb --database=default
```

And if you have the legacy acs catalogue do:

```
python manage.py syncdb --database=acs
```

The django fixtures included with this project should populate the initial database when you run the above command.

5.3.2 Restoring an existing database

Nightly backups are made on lion at:

```
/mnt/cataloguestorage1/backups/YEAR/MONTH/DAY/
```

To restore the backup do:

```
pg_restore sac_postgis_30August2010.dmp | psql sac
pg_restore acs_postgis_30August2010.dmp | psql acs
```

5.4 Setup apache (mod python way)

Note: This will be deprecated in favour of mod_wsgi (see next section)

Make sure you have mod_expires and mod_deflate installed.

The assumption is that you are using name based virtual hosts and that the catalogue will run at the root of such a virtual host. Add to you apache site config:

```
cd apache
cp apache-site-modpy.templ catlogue-modpy
```

Modify as appropriate your closed catalogue-modpy file the source tree then link it to apache.

```
sudo ln -s catlogue-modpy /etc/apache2/sites-available/catalogue-modpy
```

Also do:

```
sudo apt-get install libapache2-mod-python
```

Now deploy the site:

```
sudo a2ensite catalogue-modpy
sudo /etc/init.d/apache reload
```

5.5 Setup apache (mod_wsgi way)

The assumption is that you are using name based virtual hosts and that the catalogue will run at the root of such a virtual host. Add to you apache site config:

Modify as appropriate a copy of the apache-site-wsgi.templ file found in the apache dir in the source tree then link it to apache.

```
cd apache
cp apache-site-wsgi.templ catlogue-wsgi
```

Now create a symlink:

```
sudo ln -s catlogue-wsgi /etc/apache2/sites-available/catalogue-wsgi
```

Also do:

```
sudo apt-get install libapache2-mod-wsgi
```

Now deploy the site:

```
sudo a2ensite catalogue-wsgi
sudo /etc/init.d/apache reload
```

5.6 Copy over the ribbon

There is a ribbon image that displays in the top left corner of the site that is used to convey version numbers etc. Since this may vary from deployment to deployment, you should copy over an appropriate ribbon e.g.:

```
cp media/images/ribbon_template.png media/images/ribbon.png
```

5.7 Install GEOIP data

GeoIP is used to resolve IP addresses to Lon/Lat. This directory needs the GeoIP lite dataset in it:

```
cd geoip_data
wget http://www.maxmind.com/download/geoip/database/GeoLiteCity.dat.gz
gunzip GeoLiteCity.dat.gz‘
```

5.8 Check settings.py!

Go through settings.py (after first copying it from settings.py.templ if needed) and check all the details are consistent in that file.

5.9 Install proxy.cgi - note this will be deprecated

Some parts of this site use cross site XHttpRequests. This is not allowed in the spec (to prevent cross site scripting attacks) so to get around this you need to install a proxy cgi on the django hosting server *if the mapserver instance is on a different physical server*.

```
cd /usr/lib/cgi-bin
sudo wget -O proxy.cgi \
http://trac.openlayers.org/browser/trunk/openlayers/examples/proxy.cgi?format=raw
sudo chmod +x /usr/lib/cgi-bin/proxy.cgi
```

Once you have installed the proxy.cgi you need to configure it to tell it the list of allowed servers it can proxy for. This is to prevent it becoming an open relay on the internet. Edit /usr/lib/cgi-bin/proxy/cgi and change line 18 to look like this:

```
allowedHosts = [ '196.35.94.243','lion', ]
```

I also changed line 32 to look like this:

```
 url = fs.getvalue('url', "http://196.35.94.243")
```

so that the default proxy url is our wms server.
See http://faq.openlayers.org/proxyhost/all/ for more info...

5.10 Creating branches

Note: This section uses svn commands and should be updated to use git equivalents.

When the code gets stabilised to a certain point you should create a branch to mark that stable code base and then deploy it on the live server. To create the branch do e.g.:

```
svn cp https://196.35.94.196/svn/trunk/sac_catalogue \
       https://196.35.94.196/svn/branches/catalogue_v1_beta3
```

Where: **v1** = version 1 **beta3** = the current status of that major version

5.11 Backup of the web server

```
sudo dd if=/dev/sdb | ssh definiens4 "dd of=/cxfs/dd_backups/orasac1/orasac1_sdb_‘date +%a%d%b%Y‘.dd"
sudo dd if=/dev/sda | ssh definiens4 "dd of=/cxfs/dd_backups/orasac1/orasac1_sda_‘date +%a%d%b%Y‘.dd"
```

5.12 Creation of the ReadOnly db user

This should be done on the database server i.e. elephant

This user is required for mapserver access to some of the tables.

```
sudo su - postgres
createuser -S -D -R -l -P -E -e readonly
exit
psql sac
grant select on vw_usercart to readonly;
grant select on visit to readonly;
grant select on sensor to readonly;
\q
```

5.13 Optimal database configuration

To support the large number of recs tweak /etc/postgresql/8.3/main/postgresql.conf

```
# Changed by Tim as the sac db required more
max_fsm_pages = 500000
```

Then restart the db

```
sudo /etc/init.d/postgresql restart
```

5.14 set some file permissions

Apache user needs write access in avatars:

```
sudo chgrp www-data media/avatars
sudo chmod g+w media/avatars
```

5.15 ER Diagram

You can generate an ER diagram for the application using the django command extensions:

To generate the graph use:

```
python manage.py graph_models catalogue > docs/catalogue_diagram.dot
cat docs/catalogue_diagram.dot | dot -Tpng -o docs/catalogue_diagram.png ; \
  display docs/catalogue_diagram.png
```

5.16 SVN Ignoring files

Please read this:

http://stackoverflow.com/questions/116074/how-to-ignore-a-directory-with-svn

so that files that do not belong in svn are not shown in the status list.

5.17 Troubleshooting

5.17.1 settings.py not found

This is usually a symptom that one of the imports withing settings.py failed. Test by doing:

```
python
```

Then at the python prompt do

```
import settings
```

The error you obtain there (if any) will be more descriptive.

6 Running unit tests

A settings file for tests is available as 'settings_test.py' this settings use the faster spatialite instead of createdb.

Note You must enable the virtual environment first:

```
source ../python/bin/activate
```

6.1 Running unit tests using SQLITE backend

Before you can run the tests, you need to make sure you have pysqlite installed:

```
pip install pysqlite
```

It should have been installed during the system setup process already.

Run tests for catalogue app as:

```
$ python manage.py test catalogue --settings=settings_test
```

6.2 Running Unit tests using Postgresql

Alternatively you can use postgresql as the test database backend. Before you can run the tests you should create a template database and set some permissions on it:

```
createdb template_postgis
psql template_postgis
GRANT ALL ON geometry_columns TO PUBLIC;
GRANT ALL ON spatial_ref_sys TO PUBLIC;
\q
```

Now you can run the tests without the settings_test option and they will be executed against an autogenerated PostgreSQL database backend.

```
python manage.py test catalogue
```

7 Catalogue Reporting tools

The catalogue provides numerous interactions for users and is continually being updated with new metadata records. It is useful to produce reports that allow SANSA staff to obtain the pulse of the system. These reports cover 4 main areas:

1. Data holdings
2. Search activities
3. Visitor statistics
4. Order and tasking activities

The reports can be obtained in one of two ways:

1. Visiting the 'staff' area of the web site and selecting from the reports presented there

2. By direct email. Here staff can nominate which reports they wish to receive, and with which frequency they receive them

Note: Only 'staff' members are elegible to receive reports.
Reports sent by email will be in either rich html format, or as pdf attachments.

7.1 Order summaries

7.1.1 Order summary table

The order summary table is accessible from the **Staff -> Orders list** and for individual users from **Popular Links -> My orders**. For individual users, only their own orders will be listed. In all other respects, both tables are the same. The table contains the following headers:

When clicked, the headers will set the sort order for the table.
Above the table is a chart which displays total orders by status. For individual users, this chart shows only their own orders.
The summary report link on the summary table will return the user an on-the-fly created order summary report in pdf format as described below.

7.1.2 Order summary report

The order summary report will be sent to nominated users at chose interval of daily, weekly or monthly. It can also be generated on-the-fly from the staff admin interface.
The orders summary report contains the following information:

- How many orders have been created in the reporting month
- How many orders have been closed in the reporting month
- A break down of all open orders by status (accepted etc.)
- A break down of all open orders by age
- A break down of all orders by customer

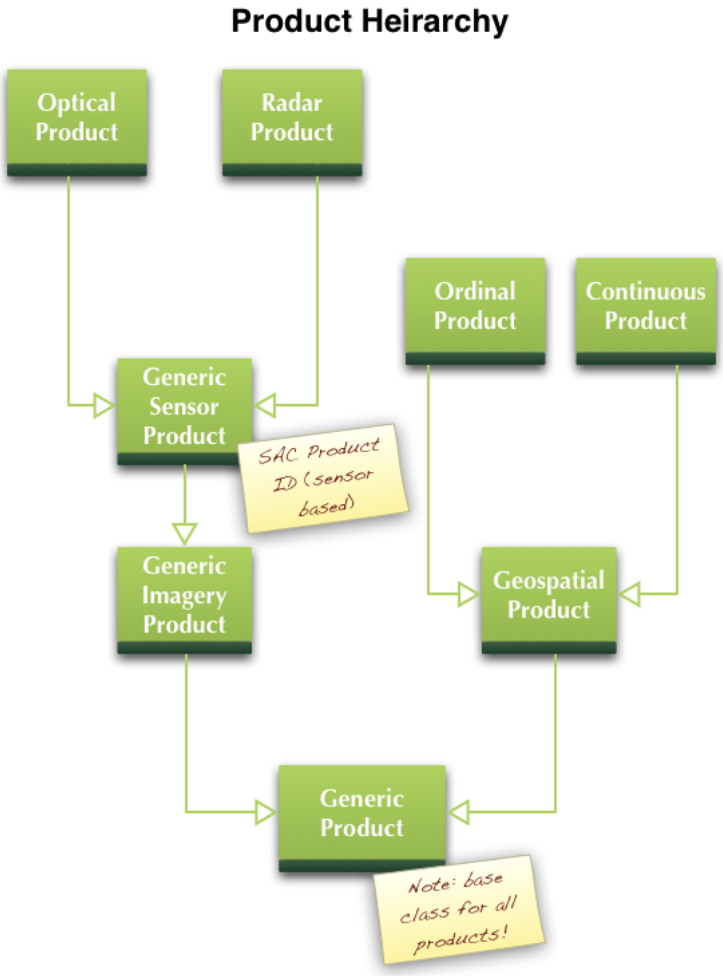
Format : pdf

8 Catalogue Schema : Products

The working unit of the catalogue is a product. A product can be any of a range of different type of geodata:

- satellite imagery
- processed imagery
- other satellite products e.g. radar data
- derived products e.g. pan sharpened imagery
- composite products (and composite derived products) e.g. mosaics
- vector and raster data of an ordinal nature (where the data is arranged in discrete classes) e.g. landcover maps
- vector and raster data of a continuous nature (where the data are arranged within a numeric scale) e.g. rainfall monitoring points

The catalogue implements a model (see figure below) that caters for these different types of product and tries to deal with them in a consistent way, and groups them in a manner that cross cutting properties and behaviours can be assigned to any given product based on the family of products to which it belongs.



This is the revised schema for version 2 of the online catelogue's product model.

In this chapter we delve into the various subtypes of product and explain the operational rules governing each type.

8.1 Generic Products

Synopsis: Abstract Base Class for all products.

Concrete or Abstract: Abstract (A product must be a subclass of Generic Product)

The generic product is the base class of all products (sensor based or derived / surveyed geospatial data). The purpose of the generic product is to define common properties applicable to **any** product regardless of type. A number of data dictionaries (as described in the next section) are used to ensure data consistency for properties relating to a product.

8.1.1 Product ID Naming Scheme

All generic products can be identified by a 'nearly unique' product id. The product id seeks to normalise the naming conventions used by different satellite operators such that a common naming scheme can be universally applied.

The naming scheme used for products depends on where the product falls in the model heirarchy. Each product type (Generic, Imagery, Sensor, Geospatial etc.) can overload the getSacProductId method in order to apply model specific logic as to how these product Ids should be assigned.

These will be discussed more fully in the sections that follow.

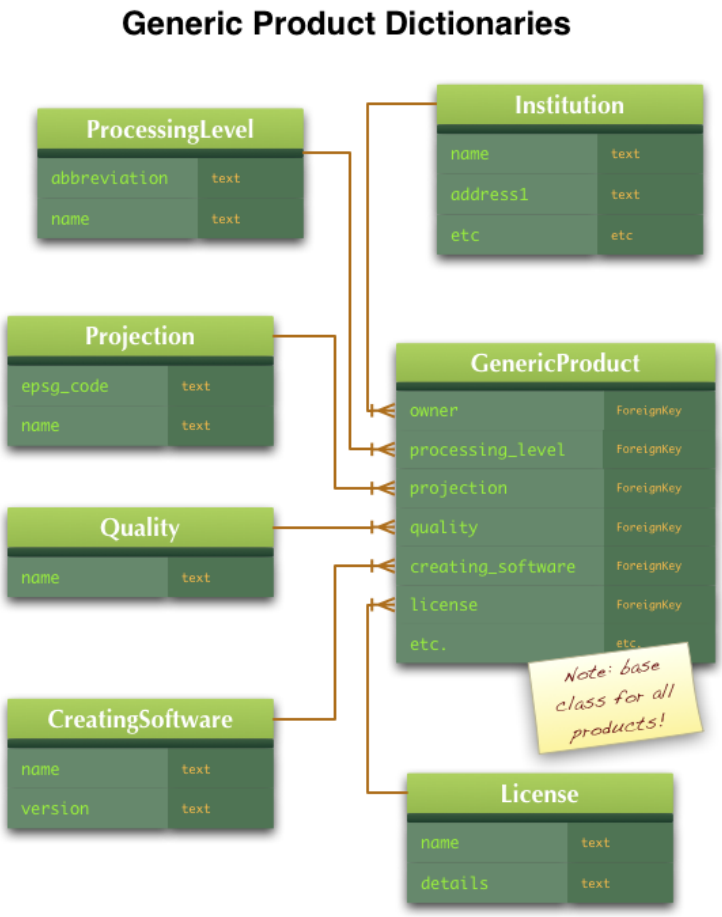
Since this is an abstract class, it has no direct naming scheme of its own.

———— **Note:** As per discussion with Wolfgang on 17 Feb 2011, the SAC Product Id will apply only to sensor based products and a different, as yet undefined, identifcation scheme will be used for GeoSpatial products.

Note: See CDSM naming scheme for vector products.

8.1.2 Dictionaries

Generic product properties that are used repeatedly are described using foreign key joins to various dictionary tables. These can be visualised in the following diagram:



These dictionaries are described in details in the sub-sections that follow.

Institution

The institution (linked to the **GenericProduct** table on the owner field) indicates the organisation from which the product can be obtained.

As at the time of writing this document, only one record exists in this table, and all new products are assigned to this institution:

ID	Name	Address1	Address2	Address3	Post Code
1	Satellite Applications Centre	Hartebeeshoek	Gauteng	South Africa	0000

Note: Wolfgang verify that we won't be storing the original data owner (e.g. Spot Image) here.

Processing Level

Products may have been processed by software to improve the product. For example, a level 1a product is a 'raw' image with no georeferencing, format conversion etc. In order for a product to be usable by mainstream users, it generally needs to be converted into a popular image format (e.g. geotiff), georeferenced (so that it appears in the correct place on a map), orthorectified (to adjust the image based on distortions introduced by terrain variance) and so on.

Processing levels are expressed as a four letter code e.g.:

```
L1Aa
```

This code can be deconstructed as:

- **L** Abbreviation for 'level'
- [**1-4**] A numeral representing the major level wher
  - **1** Raw imagery
  - **2** Unreferenced imagery in a common format e.g. tiff
  - **3** Rectified and imagery cleaned for atmospheric disturbance, lens irregularities etc.
  - **4** Derived products
- [**A-C**] A single upper case character representing product class (derivative,raster, vector)
- [**a-z**] A single lower case character representing class-type (see below)

Note: The 'L' prefix is not stored in the database tables.
Some commonly used codes include:

Code	Description
2A/B	(L1G)
3Aa	(L1T) Orthorectified DN values
3Ab	At sensor/TOA reflectance
3Ac	Atmospherically corrected/TOC reflectance
3At	Topographic correction
4A*	Derivatives/products
4B*	Pixel-based classification
4C*	Vector/object classification

Some common values for class type include:

Code	Description
a	Ancillary data eg. Relief shadow, hydrology
b	Bare or built-up indices
m	Maths transformation
s	Statistical calculations
t	Texture
v	Vegetation indices
w	Water/moisture indices
x	Time-series or metrics
f	Spectral Rule based Features including indices
r	L1 Spectral Rule based layers
c	L2 Spectral Rule based layers classification spectral categories

The following processing levels are listed in the database.

ID	Abbreviation	Name
1	2A	Level 2A
2	1A	Level 1A
3	1Ab	Level 1Ab
4	1Aa	Level 1Aa
12	3Aa	Level 3Aa
13	3Ab	Level 3Ab
14	3Ac	Level 3Ac
15	3Ad	Level 3Ad
16	3Ba	Level 3Ba
17	3Bb	Level 3Bb
18	4	Level 4

Note: These should be updated to include a proper description with each in a new description col. TS

Projection

The projection (or more accurately the coordinate reference system (CRS)) model contains a dictionary of CRS identifiers and human readable names. The identifiers are expressed in the numbering system of the European Petroleum Survey Group (EPSG). The list included is not comprehensive - at the time of writing it contains only 84 or the *circa* 3000+ entries available in the official EPSG CRS list.

The EPSG name is a more user friendly and easily recognisable representation of the EPSG code. For reference, an extract of the entries is provided in the table below:

ID	Epsg Code	Name
1	32737	UTM37S
2	32733	UTM33S
3	32738	UTM38S
4	32734	UTM34S
5	32732	UTM32S
6	32735	UTM35S
7	32629	UTM29N
8	32731	UTM31S

Quality

The quality is intended to provide a well defined dictionary of terms for qualitative assessment of products. Different product vendors use different schemes for describing product quality.

For example Spot Image describes quality in terms of an imaginary grid overlaid (and bisecting the image into equal sized units). Each grid cell is then given a ranking e.g. "AABBAAAABB".

Currently only one entry exists:

ID	Name
1	Unknown

This presence of a quality indicator is mandatory for GenericProduct, but the fact that all records are currently assigned a ranking of 'Unknown' makes this attribute largely meaningless.

Note: Wolfgang - we need to define a standard of quality description and a strategy for populating existing and new records with an appropriate quality indicator.

Creating Software

It is useful in understanding a dataset to know which software package was used to create it. At time of writing this document, only two software packages are available:

ID	Name	Version
1	Unknown	None
2	SARMES1	Sarmes1

Sarmes2 will be added to this list in the future, and other additional packages as needed.

Note: Wolfgang - do we need to include other software here, and if so which products should have this applied?

License

Each product should have a license associated with it. The license will detail any restrictions on redistribution or useage that applies for the product.

The following licenses are defined and all products have been assigned one of these licenses.

ID	Name	Details	Type
1	SAC Commercial License	SAC Commercial License	
2	SAC Free License	SAC Free License	
3	SAC Partner License	SAC Partner License	

License Type Enumeration: The license type is an enumerated list (managed directly in the model rather than in a separate dictionary via a foreign key constraint). The reason for this is that application logic specific to the license type is implemented (for example to determine if a product can be freely distributed to a client).

Each license in the system is allocated a type. The following types and their meanings are defined:

Free data, government license or commercial license or any.

ID	Type	Description
1	Free	Can be freely shared and redistributed without restriction
2	Government License	Applies to products that can be freely redistributed to government departments.
3	Commercial	Applies to product that can only be commercially distributed

The license type is determined on a sensor by sensor, product by product basis. The following rules hold true:

1. SPOT data are all under SAC Partner license
2. SAC-C, Sumbandilasat and CBERS are all under SAC Free License
3. When not explicitly defined, all products should be assigned the SAC Commercial License

Note: Wolfgang to define any further rules

Note: The allocation of license may not always reflect the cost of the data

- where substantial processing has been requested by a user, SAC may charge a processing fee. The system currently makes no accommodation for calculation of such 'value added' fees.

Note: For the Government license type The User profile for this catalogue includes a field SacUserProfile::strategic_partner which is a boolean indicating if the users is registered as a SAC Strategic Partner employee and thus granted unfettered access to certain products (e.g. Spot imagery). Where the user is not a Government employee, the product license should be considered to be commercial.

Note: Wolfgang - we need to define more completely the SAC License, or several variants of it, and add any other licenses that may apply. We would also need rules descibing how to select products which should have which license applied.

Note: Wolfgang - does it make sense to have an 'any' license?

8.2 Generic Imagery Products

Synopsis: Base Class for *imagery* products.

Concrete or Abstract: Concrete (instances of this class can be created and stored).

GenericImageryProduct is the model that all sensor based products inherit from. In addition, concrete instances of **GenericImageryProduct** are used to lodge sensor based aggregate data. For example when an image is created that is a combination of a 'J' and a 'T' image, we can no longer canonically state which acquisition mode etc was used for that image. In this case the DAG (Directed Acyclical Graph) implementation will be used to provide backpointers to the original images used to create this record (and those backpointers will be to Sensor based product records).

8.2.1 Product ID Naming Scheme

Generic imagery products do not have an associated Mission, Sensor, Sensor Type or Acquisition Mode, and thus the naming system differers from final (having no ancestors) sensor based products.

Note: Wolfgang to define or we must devise something

8.2.2 Imagery product Properties

A GenericImageryProduct extends the generic product model with these properties:

- geometric_resolution
- geometric_resolution_x
- geometric_resolution_y

8.2.3 Imagery Product Aggregation Rules

In the DAG (Directed Acyclical Graph) that maps relationships between products and their downstream constituent products, ImageryProducts can be made of:

- Generic Imagery Products
- Generic Sensor Products
- Optical Products
- Radar Products
- Geospatial Products and subclasses of Geospatial products

Note: Self referencing is not allowed. That is, a Imagery product may not include itself in any leaf node.

Note: Generic Imagery Products will always be composite (derived from one or more other products).

8.2.4 Dictionaries

No additional dictionaries are introduced with this class.

8.3 Generic Sensor based products

Synopsis: Base Class for *Sensor* products.

Concrete or Abstract: Abstract (instances cannot be directly created and stored).

Generic sensor product is an Abstract Base Class that all other sensor based products inherit from. It inherits from Generic Imagery product.

8.3.1 Product ID Naming Scheme

The following scheme is used for assigning product id's for sensor based products:

Single scene file names:
`SSS_sss_ttt_mmmm_pppp_ps_rrrr_rs_yymmdd_hhmmss_LLLL_PPPPPP`
e.g. L7-_ETM_HRF_SAM-_168-_00_077-_010530_–_L3Ab_UTM36S

Composite files -mosaics:
`QQQQQQ_SSS_sss_ttt_mmmm_pppp_ps_rrrr_rs_yymmdd_hhmmss_LBLL_PPPPPP`
Use central scene for date and time

Composite files - time series:
`MT_yymmdd_yymmdd_SSS_sss_ttt_mmmm_pppp_ps_rrrr_rs_yymmdd_hhmmss_LBLL_PPPPPP`
Use central time scene for date and time

The following key can be used to decode the above:

Code	Description
SSS	satellite (mission) name e.g. L5-; S5-
sss	sensor (mission sensor) e.g. TM-; ETM
ttt	type (sensor type) eg. HRF; HPN; HPM
mmmm	bumper (acquisition) mode eg. SAM- BUF-
pppp	path
ps	path shift
rrrr	row
rs	row span
yymmdd	date
hhmmss	time
LLLL	processing Level code eg. L2A; L4Ab
PPPPPP	Projection eg. UTM35S; LATLON; ORBIT-
QQQQQQ	1:50000 topographic map name eg 3425CD
MT_yymmdd_yymmdd	multi-temporal time span: start date to end date

Note: Elements Mission(SSS), MissionSensor(sss), SensorType(ttt) and Acquisition-Mode (mmmm) are compulsory for GenericSensorProducts. Where these are not explicitly defined, they will be implicitly defined. That is to say, if no Sensor has been defined for a Mission, it will cause the creation and assignment of an implicit MissionSensor orbject named after the sensor. For example: A new product for fictitious sensor 'FOO' is imported. No sensor type is defined, so a default sensor type called 'FOO' is defined. This same logic applies to SensorType and Acquisition Mode. Another example: Mission L5 exists, Sensor MS exists, no sensor type exists so a default type of 'MS' is created and consequently a default acquisition mode of MS os also created. See 'placeholder' clauses in description that follow.

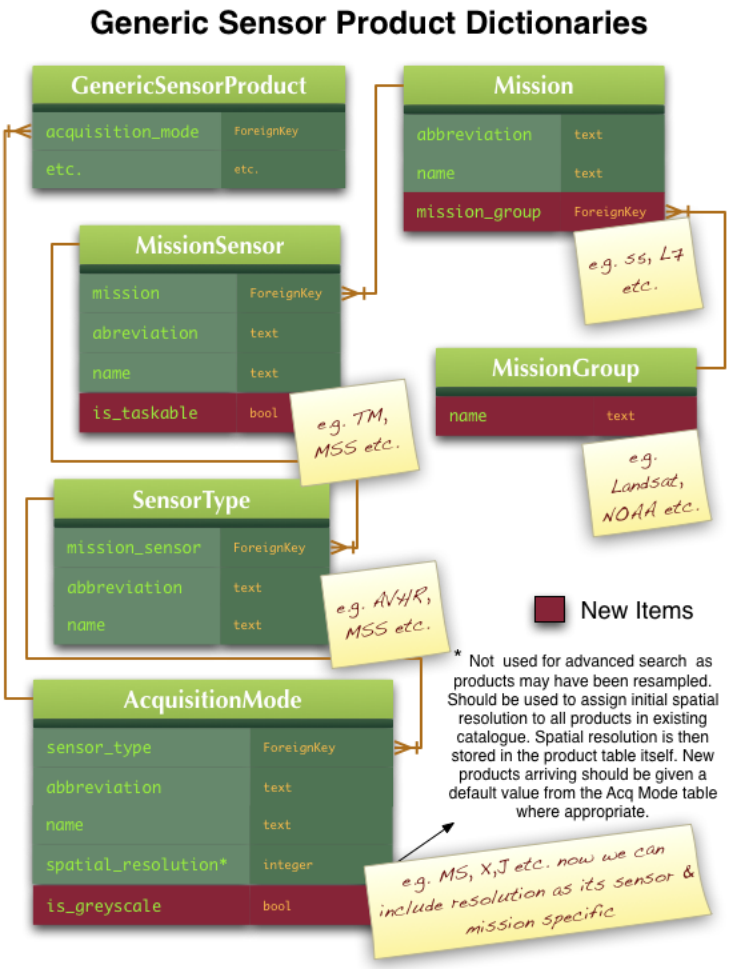
Where Row (rrrr) and Path (pppp) are not know or not applicable for a sensor, the word 'None' will be written out in the product id to differentiate from 0000 which may be a valid row or path.

8.3.2 Generic Sensor Product Aggregation Rules

Since this is an abstract class it may not be a node in a product aggregation tree.

8.3.3 Sensor Product Dictionaries

Several domain lists are implemented for sensor based products. These can be visualised in the following diagram:



These dictionaries and their interrelationships are described in more detail in the text below.

Mission

A mission is the name for the particular space vechicle on board of which one or more sensors are deployed. The catalogue hosts metadata entry for a number of different sensors - at time of writing this list looked like the table below.

ID	Abbreviation	Name
1	N14	Noaa 14
2	N16	Noaa 16
3	N11	Noaa 11
4	N9	Noaa 9
5	N17	Noaa 17
6	N12	Noaa 12
7	N15	Noaa 15
8	E2	E-Ers 2
9	E1	E-Ers 1
10	L5	Landsat 5
11	L7	Landsat 7
12	L2	Landsat 2
13	L3	Landsat 3
14	L4	Landsat 4
15	S2	Spot 2
16	S4	Spot 4
17	S1	Spot 1
18	S5	Spot 5
19	ZA2	Sumbandilasat
20	C2B	CBERS
21	S-C	SAC-C

Note: Wolfgang we are using ZA2 instead of SS here.
Note: RE - RapidEye needs to be added to this list

Mission Sensors

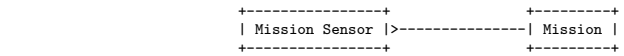
On board each space vehicle receiving EO data will be one or more sensors. Although the sensor may be nominally the same between two different missions (e.g. MSS), the specific properties of these sensors will vary between missions, and even between identical sensors on the same mission. A sensor is basically a camera or other equipment capable of performing distance observation of the earth.

ID	Abbreviation	Name	Description	Has Data	Mission
1	AVH	NOAA AVHRR		t	NOAA
2	AMI	ERS AMI SAR		t	ERS
3	TM	Landsat 4 TM		t	L4
4	TM	Landsat 5 TM		t	L5
5	MSS	Landsat 1 MSS		t	L1
6	MSS	Landsat 2 MSS		t	L2
7	MSS	Landsat 3 MSS		t	L3
8	MSS	Landsat 4 MSS		t	L4
9	MSS	Landsat 5 MSS		t	L5
10	ETM	Landsat 7 ETM+		t	L7
11	Xs1	Spot 1 HRV Xs		t	S1
12	Xs2	Spot 2 HRV Xs		t	S2
13	Xs3	Spot 3 HRV Xs		t	S3
14	Xi	Spot 4 G,R,NIR,SWIR		t	S4
15	M	Spot 4 Pan		t	S4
16	Pan	Spot 1 HRV Pan		t	S1
17	Pan	Spot 2 HRV Pan		t	S2
18	Pan	Spot 3 HRV Pan		t	S3
19	HRG	Spot 5 HRG	Spot 5 HRG	t	S5
20	CCD	CBERS CCD		t	C2B
21	MRS	SACC MRS		t	S-C
22	SMS	Sumbandilasat MSS		t	ZA2

Note: Ale, I would like to get rid of the has data field if poss. It is used to restrict the display of sensors to users to only those with data associated to them. Is there a cleaner way to do it? At mininimum we need to automate updating this field as data is added to and removed from the system.

Sensors TM - Thematic Mapper ETM - Enhanced thematic mapper MSS Multi-spectral

Note: A one-to-many relationship will be created between mission sensor entities and their related mission - as illustrated here:



Relationship in plain english: Each mission can have one or more mission sensors associated with it. Each mission sensor shall be associated to only one mission.“

The Abbreviation will **not** be unique per sensor. Note that the 4 letter name space does not allow for many permutations and readers of the abbreviation should do it in context of a specific mission, sensor type etc. (for example by always presenting the mission abbreviation at the same time).

In some cases, no specific mission sensors will exist for a mission. In these cases, a placeholder mission sensor should be created, named directly after the mission e.g.

ID	Abbreviation	Name	Description	Has Data	Mission
1	ERS	ERS	ERS	t	ERS

Above assumes the pre-existance of a mission 'ERS'.

Sensor Types

The sensor type describes a mission sensor. 'High end' satellites may use a custom sensor type with its own specific properties. 'Cheaper' satellites may use simple CCD (charge coupled device) style sensors.

ID	Abbreviation	Name	Mission Sensor
1	AVHR	Advanced Very High Resolution Radiometer	
2	AMI	AMI	
3	MST	Multispectral + Thermal	
4	CAM2	Spot Camera 2	
5	CAM1	Spot Camera 1	
6	R3B	R3B	
7	MSS	Multispectral	
8	RT	RT	

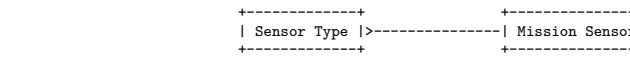
Note: Wolfgang please populate the related mission sensor(s) per id above.

Each mission sensor should have at least one sensor type associated with it. When a mission sensor exists with no associated sensor type, a default sensor type matching the mission sensor abbreviation should be created e.g.

ID	Abbreviation	Name	Mission Sensor
7	MSS	Multispectral - Landsat 1	Landsat 1 MSS
8	MSS	Multispectral - Landsat 2	Landsat 2 MSS
9	MSS	Multispectral - Landsat 3	Landsat 3 MSS
10	MSS	Multispectral - Landsat 4	Landsat 4 MSS
11	MSS	Multispectral - Landsat 5	Landsat 5 MSS

The Abbreviation will **not** be unique per sensor type. Note that the 4 letter name space does not allow for many permutations and readers of the abbreviation should do it in context of a specific mission-sensor, acquisition mode etc. (for example by always presenting the mission abbreviation at the same time).

Note: A one-to-many relationship will be created between mission sensor entities and their related mission - as illustrated here:



Relationship in plain english: Each mission sensor can have one or more sensor types associated with it. Each sensor type shall be associated to only one sensor.“

Acquisition Modes

Each sensor can operate in one or more modes. Thus there the list of acquisition modes should include at least one entry per sensor type. Where such an entry does not exist, a default one (named after the sensor type) shall be created. Acquisition modes table example:

ID	Abbreviation	Name	Geom Resolution	Band Count	Sensor Type
1	MS	Multispectral	0	0	
2	VV	Vertical / Vertical Polarisation	0	0	
3	HRT	Multispectral and Thermal	0	0	
4	X	X	0	0	
5	I	I	0	0	
6	M	M	0	0	
7	P	P	0	0	
8	J	Multispectral	0	0	
9	B	Panchromatic	0	0	
10	A	Panchromatic	0	0	
11	FMC4	FMC4	0	0	
12	3BG	3BG	0	0	
13	5BF	5BF	0	0	
14	3BP	3BP	0	0	
15	HR	HR	0	0	

Note: Wolfgang to populate sensor type. Entries should be duplicated for each sensor type that has that mode e.g.

ID	Abbreviation	Name	Geom Resolution	Band Count	Sensor Type
1	MS	Multispectral - Landsat 1	0	0	Multispectral - Landsat 1
2	MS	Multispectral - Landsat 2	0	0	Multispectral - Landsat 2

Todo: Check this list from Wolfgang is represented:

Mode	Description
HRF	Multi-spectral bands
HPN	Panchromatic bands
HTM	Thermal bands
HPM	Pan-sharpened multi-spectral

Note: A one-to-many relationship will be created between mission sensor entities and their related mission - as illustrated here:

Relationship in plain english: Each sensor type can have one or more acquisition mode associated with it. Each acquisition mode shall be associated to only one sensor type.“

The Abbreviation will **not** be unique per acquisition mode. Note that the 4 letter name space does not allow for many permutations and readers of the abbreviation should do it in context of a specific sensor type. (for example by always presenting the sensor type, mission type and mission abbreviations at the same time).

Note: The geometric resolution and band count columns in this table need to be populated by Wolfgang.

Logic Rules: The additional columns in the acquisition mode are used to provide implicit values for products if they are created without implicit values. In particular:

1. The geometric resolution will be used to populate the GenericSensorProduct fields of geometric_resolution, geometric_resolution_x and geometric_resolution_y. This is implemented by first populating the geometric_resolution field (NULL) when assigning acquisition mode, and then assigning the same value to both geometric_resolution_x and geometric_resolution_y.

2. The band_count in acquisition mode will be assigned to the GenericSensorProduct.band_count (formerly called spectral_resolution) if NULL at the moment of assignment.

Mission Group

The mission group model will be an addition to the schema that will allow virtual groupings of mission sensors. In simple search and other places designated by the client, mission groups will be used to select a family of missions (satellites) to search on e.g. all Landsat missions.

Relationship in plain english: Each mission group can have one or more missions associated with it. Each mission shall be associated to only one mission group.“

The mission group should be populated as follows:

When a new mission is added, it should always be assigned to an existing mission group (with the mission group being created first if needed).

8.3.4 Notes on the proposed schema changes

The proposed schema change will bring about the following advantages:

- less attributes stored on generic sensor (simplification is always good)
- easier to understand the relationship between these entities
- remove the risk of ambigious entries (e.g. MSS applying to multiple different sensors)
- we can now effectively store resolution on acquisition table as its unambiguous as to which sensor & mission it applies
- product id 'drill down searches will be more efficient

Note: We need to get the mappings from Wolfgang for which mission sensors are associated with each mission etc.

Because the schema changes introduce a strict heirarchy and also introduce a requirement that acquisition mode through to mission be defined for a sensor based product, assigning these entities to derived products will not be meaningful. Because of this composite products (e.g. a pan sharpened SPOT image) will not be modelled under generic sensor products but rather belong to a sub class GenericImageryProduct.

8.3.5 Resolving the metadata to explicit records

The input metadata we receive will be ambiguous for acquisition mode, sensor type, mission sensor. It is only with the presence of a mission abbreviation that these can be correctly resolved. For example:

`L7-_ETM_HRF_SAM-_168-_00_077-_010530_------_L3Ab_UTM36S`

`SSS_sss_ttt_mmmm_pppp_ps_rrrr_rs_yymmdd_hhmmss_LLLL_PPPPPP`

Code	Description
SSS	satellite (mission) name e.g. L5-; S5-
sss	sensor (mission sensor) e.g. TM-; ETM
ttt	type (sensor type) eg. HRF; HPN; HPM
mmmm	bumper (acquisition) mode eg. SAM- BUF-

So we can see from this example, we have the following:
||Satellite | Mission Sensor | Sensor Type | Acquisition Mode |

L7	ETM	HRF	SAM

In cases where the entries for these dictionary terms do not exist, new records should be added to the tables using the following logic:

1. Add `L7` to the mission table and note the PKEY of the new record
2. Add abbreviation `ETM`, description `ETM:Landsat 7`, mission `PKEY from above` to the mission sensor table and note the PKEY of the new record
3. Add abbreviation " HRF, `description ''HRF:ETM:Landsat 7`, mission_sensor `PKEY from above` to the sensor type table and note the PKEY of the new record
4. Add abbreviation `SAM`, description `SAM:HRF:ETM:Landsat 7`, sensor_type `PKEY from above` to the acquisition mode table.

8.4 Optical products

Synopsis: Vector and Raster products where data are grouped into discrete classes.

Concrete or Abstract: Concrete

Optical products are a specialisation of Generic Sensor Products. An Optical Product is a concrete class (i.e. one that should not be treated as abstract). The Optical Product model is used to represent any sensor originated product that has been taken using an optical sensor. It may cover non-visible parts of the spectrum and consist of one or more bands, each covering a different part of the spectrum. Generally the bands (in multiple band images) are co-aligned - meaning the cover the same geographical footprint. In some cases however, the bands are offset from each other, creating an opportunity to super-sample the image and improve its native resolution.

Optical products are end nodes in the product heirarchy - they do not have any further specialisations.

8.4.1 Optical Product Properties

8.4.2 Product ID Naming Scheme

8.4.3 Optical Product Aggregation Rules

In the DAG (Directed Acyclical Graph) that maps relationships between products and their downstream constituent products, **OpticalProducts** may **not** themselves be aggregates. This is because each sensor product has an explicit acquisition mode, sensor type etc. and such relationships are not mappable for aggregate products. OpticalProducts can however participate in aggregations. In the case that you have have two **OpticalProducts** forming a new image, the new image should be modelled as a **GenericImageryProduct**.

8.5 Radar Products

8.5.1 Radar Product Aggregation Rules

In the DAG (Directed Acyclical Graph) that maps relationships between products and their downstream constituent products, **RadarlProducts** may **not** be aggregates. This is because each sensor product has an explicit acquisition mode, sensor type etc. and such relationships are not mappable for aggregate products. In the case that you have have two **RadarlProducts** forming a new image, the new image should be modelled as a **GenericImageryProduct**.

8.5.2 Product ID Naming Scheme

Follows the same scheme as defined in OpticalProduct documentation.

8.6 Geospatial Products

Synopsis: Level 4 products derived from imagery or by direct earth measurement (e.g. by conducting a survey with GPS).

Concrete or Abstract: GeoSpatial products are pure abstract (they cannot exist on their own, only as a subclass).

Geospatial Products incorporate all level 4 products. Geospatial products are abstract representations of features of the earths surface - as opposed to Imagery Products which are some form of obervation of the earth service.

8.6.1 Geospatial Product Properties

All geospatial products share the following properties:

- description
- equivalent_scale
- temporal_extent_start
- temporal_extent_end
- region_type
- region
- primary_topic
- tags

Properties from GenericProduct are also inherited.

The description property defines a generalised description of the product. It will be incorporated into the product abstract that is autogenerated when subclasses re-implement GenericProduct::abstract(). The description is free-form text that describes the product and any special information relating to it.

In the case of geospatial products, their spatial resolution is expressed as equivalent scale (after MD_Metadata > MD_DataIdentification.spatialResolution > MD_Resolution.equivalentScale) from the ISO19115 specification. Only the denominator is store so, for example, a value of 50000 indicates the product is suitable for use at 1:50000 scale or smaller.

The temporal extent maps to the gmd::EX_TemporalExtent element in the ISO_19136 standard. e.g.

```
<gmd:temporalElement>
  <gmd:EX_TemporalExtent>
    <gmd:extent>
      <gml:TimePeriod gml:id=\"T1\">
        <gml:beginPosition>2008-01-01</gml:beginPosition>
        <gml:endPosition>2008-03-31</gml:endPosition>
      </gml:TimePeriod>
    </gmd:extent>
  </gmd:EX_TemporalExtent>
</gmd:temporalElement>
```

Note: The GenericSensorProduct includes 367 GenericSensorProduct::product_acquisition_start and GenericSensorProduct::product_acquisition_end fields but their semantics are different - they define the actual imagery acquisition period, whereas in Level 4 / GeoSpatial products the temporal_extent_start and temporal_extent_end describe the complete date range of data represented within the dataset.

The **region** and **region_type** fields are used to define the geographic region of interest for this dataset. When represented using ISO Metadata, the region will be reflected in an EX_GeographicDescription element. e.g.

```
<xs:complexType name="EX_GeographicDescription_PropertyType">
  <xs:sequence minOccurs="0">
    <xs:element ref="gmd:EX_GeographicDescription"/>
  </xs:sequence>
  <xs:attributeGroup ref="gco:ObjectReference"/>
  <xs:attribute ref="gco:nilReason"/>
</xs:complexType
```

Note that although the ISO19139 specification allows a sequence (i.e. multiple geography description elements), our schema only accommodates a singly geographic place. As such GeoSpatiaProducts are assigned the named place that is nearest to the centroid of the product geometry, unless they have been manualy assigned. In order to assign types 1-5 (see dictionaries section below), manual intervention must be made.

Primary topic is the main categorisation for this dataset. Additional tags can be assigned to the dataset via the tags field (see below). The topic will be used when assembling the SAC product id for GeoSpatial products.

Tags are used to provide keyword based descriptive information for the product. Example tags might be: landcover, roads, trig beacons etc.

8.6.2 Product ID Naming Scheme

The following scheme will be used when allocating product ID's to GeoSpatial products:

Field	Description
TTT	Type, ORD for Ordinal products, CON for continuous products
TOPIC	10 character abbreviation for topic, hyphen padded e.g. ROADS–
yymmdd	temporal extent start date
yymmdd	temporal extent end date
LLLL	processing Level code eg. L2A; L4Ab
EPSG	EPSG Code for coordinate reference system of dataset
REG	Region name

This a sample GeoSpatial product ID may look like this:

```
ORD_LANDUSE---_110101_11_02_11_L4Ab_4326_ZA
```

8.6.3 Geospatial Product Aggregation Rules

Geospatial products may be aggregates. That is their lineage may reflect derivation from one or more other products.

8.6.4 Geospatial Product Dictionaries

The following dictionaries are implemented to support

Region Type

A region type can be one of:

ID	Type
1	Global
2	Continent e.g Africa
3	Region e.g. SADC
4	Country e.g. ZA
5	State e.g. WCAPE
6	Town or City
7	Local area - nearest named place

Regions

For regions, a dictionary based on GeoNames provides an exhaustive list of local areas and towns.

The region model looks like this:

ID	Type	Name	Long Name
1	1	Global	Global
2	2	Africa	African Continent
3	3	SADC	Southern African Development Community
4	4	ZA	South Africa
5	5	GP	Gauteng Province
6	6	Pretoria	Pretoria
7	7	Mamelodi	Mamelodi

8.7 Ordinal Products

Synopsis: Vector and Raster products where data are grouped into discrete classes.

Concrete or Abstract: Concrete

8.7.1 Ordinal Product Properties

For ordinal we define thematic accuracy by means of the following fields:

- class count
- confusion matrix
- kappa score

-

8.7.2 Product ID Naming Scheme

The product ID for Ordinal products is described in the GeospatialProduct model description. Ordinal products are prefixed with the string 'ORD'.

8.7.3 Ordinal Product Aggregation Rules

Ordingal products may be part of aggregations and their lineage may include aggregations.

8.7.4 Ordinal Product Dictionaries

No additional dictionaries are introduced by the ordinal product model.

8.8 Continuous Products

Synopsis: Vector and Raster products where data are *not* grouped into discrete classes, but rather along a continuous value range.
Concrete or Abstract: Concrete

8.8.1 Continuous Product Properties

For Continuous data there are two additional fields:

- range_min
- range_max
- unit

  The range_min is the smallest value in the

8.8.2 Product ID Naming Scheme

The product ID for Ordinal products is described in the GeospatialProduct model description. Continuous products are prefixed with the string 'CON'.

8.8.3 Continuous Product Aggregation Rules

Continuous products may be part of aggregations and their lineage may include aggregations.

8.8.4 Continuous Product Dictionaries

Unit - a look up table storing measurement units that can be present

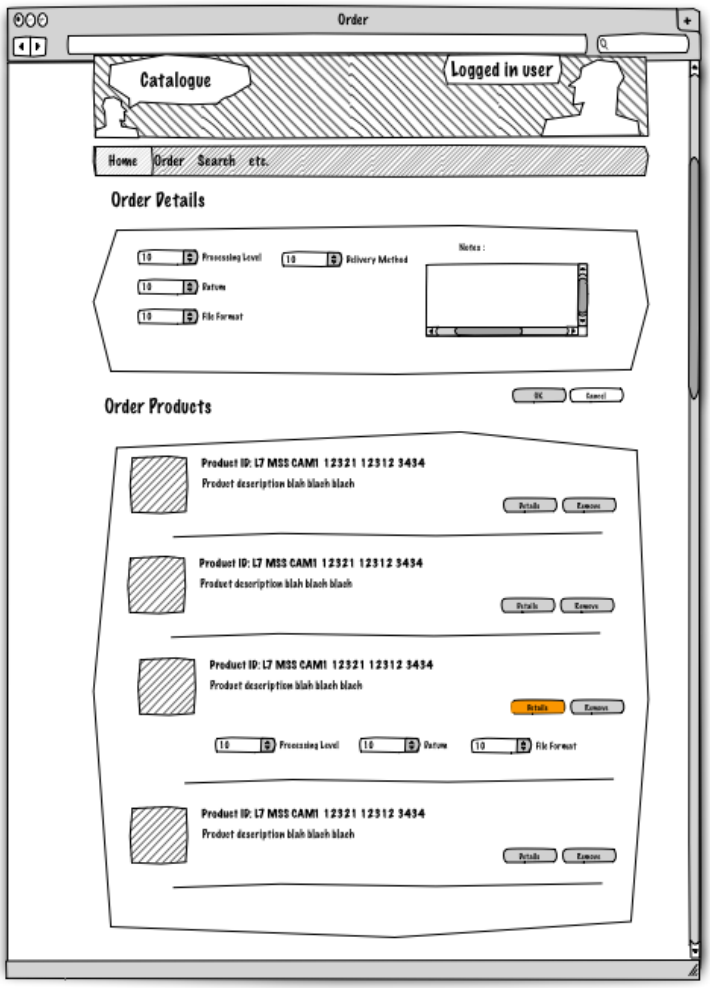
8.8.5 Dictionaries

9 Catalogue Schema : Orders

Note: Before reading this section please read & understand the products schema notes.

The purpose of the order application logic is to allow a user to easily order one or more products. Since each order can consist of one or more products, and each product can be included in many orders, a 'many to many' relationship exists between products and orders. This is realised via the **SearchRecord** model. This SearchRecord model is also described in the Search section. As a recap, the process of creating SearchRecords is achived by carrying out a search. Initially when a search is carried out, SearchRecords are transient - that is, they are not persisted in the databse. Once a user adds a SearchRecord (representing a single product) to their basket, the SearchRecord is assigned a User id and persisted in the database. A users basket or cart is thus the collection of SearchRecords owned by them but that have no Order allocated to them.

At the point of deciding to proceed with an order, the user is directed to the Order page, a wireframe for which is shown below.



The purpose of the order page is to allow the user to specify details relating to their intended order and to proceed with the order initiation thereafter. Two levels of customisation are allowed for when creating an order:

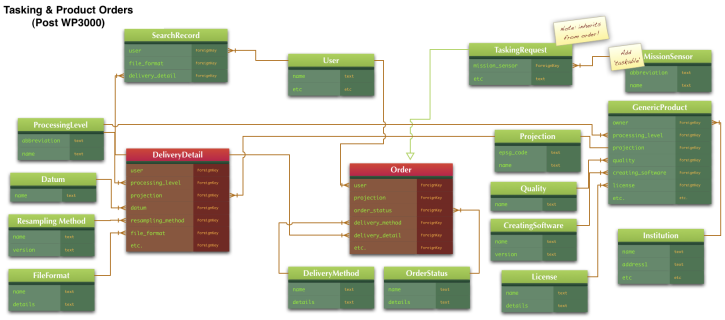
1. options that are global to the order can be defined
2. options that are specific to each product within the order can be defined (overriding the global options where applicable)

The specific global and options are specified in the sections that follow. The global versus option specification process is also illustrated in the wireframe provided above.

Note: The ordering system does not (yet) implement an e-commerce system and the processing of orders is not yet automated. When an order is placed for a product, the notifications to operators are automatic, but the process of fulfilling the orders remains operator controlled.

9.1 Schema

The schema relating to product orders can be summarised as follows:



This schema will be explained in more detail in the sections that follow.
We can express the above diagram verbally:

- Orders are instances of the Order model
- Each order has one or more SearchRecords associated with it
- Each order has a DeliveryDetail record associated with it
- Each SearchRecord *optionally* has a DeliveryDetail record associated with it
- An Order is owned by a User
- Each SearchRecord is owned by a User (which should match the Order owner)
- The DeliveryDetail record associated with an Order encapsulate the Projection, Datum, Processing level, Resampling Method and File Format that **all** the products of that order should be delivered in.
- The DeliveryDetail record associated with an SearchRecord encapsulate the Projection, Datum, Processing level, Resampling Method and File Format that the **specific product** associated with that SearchRecord should be delivered in.
- Each order has a current order status and history of OrderStatus records.
- Each order has a deliver method associated with it (e.g. ftp, portable disc, dvd etc.)

9.2 Operational notes

The following operation notes should be understood for the order system:

9.2.1 General Users

- In order for a user to initiate an order, they must have completed the details of their personal profile. If this is not the case, they will first be redirected to their profile page where they can complete their details. On completion of their details, they are returned to the order page.
- The user can remove products from the order at the point of making the order. However if he removes the last product from the order, his basket becomes empty and no order should be able to be placed.
- The user cannot edit the order after it has been placed.
- The user can request an order be cancelled after it has been placed, but should be advised that their may be a fee payable if products have already been procured.
- The user can see a list of the orders they have made and their statuses. Clicking on a specific order will take them to a detailed view of the order where they can view its current status and review the products associated with that order, and the history of status changes made for that order.
- The list of orders visible to a user is restricted to those owned by that user.
- The user's address and other contact details are shown on the order form, with a link that will take them to their profile editor so that they can update their details. After updating their details they are returned to the order.
- The user's avatar is shown on the order so that the process feels personalised to them.
- For each status change to an order (including the initial placement of the order itself), the user will recieve an email which will be provided in both plain text and html format (falling back to plain text if html is not supported by the email client being used). The email will be visually consistent with the order page and include links to products that are available for immediate download (see below).

9.2.2 SAC Staff

- SAC Staff can view a master list of all orders and filter / sort those orders by status, owner etc.
- Staff can view any order and as appropriate adjust it's status (e.g. open, in process, cancelled etc).
- No status change will be accepted without keeping a log of who made the change and what the reason was for the change.
- Each status change is timestamped so that a proper audit trail can be established.

- The order comments system should be the primary communication mechanism with the client, as it will provide a thorough audit trail.

9.3 Instant product delivery

In some cases products are available for instant download. The indicator for this is a populated GenericProduct::local_storage_path field which points to a valid resource on disk.

Where products originate from DIMS, an extract request will be made via the Ordering Service for Earth Observation products (OS4EO) implemented on the DIMS server.

In cases where instant delivery is made, download links will be included in notification emails and in the order summary visible to operators.

In the case of DUMS OS4EO extracted products, these will be flushed after a period of 1 week from the server file system and the user advised to this fact when this happens.

9.4 Filtering of CRS's

The projections list can be long and irrelevant for many images if the complete list is shown. Thus the following logic applies:

- For single product properties, when listing utm zones for the projection combo, only the zone in which the image centroid falls, and the two zones to either side of that are shown.
- For Order records, only utm zones that intersect with the centroids of one or more of the products listed in the order should be listed, along with the two adjacent zones on either side of each image.
- EPSG:4326 Geographic is always listed
- EPSG:900913 Google Mercator is always listed

9.5 Datum

Currently only one datum is supported for order requests - WGS84:

id	name
1	WGS84

Until additional datum options are available, the datum is not shown to end users since providing a datum choice selection with only a single entry is superfluous.

9.6 Processing Levels

A user can request that a product be delivered at a different processing level to the one recorded for the project. For example a user can request that an image at L1A be supplied as a L3Aa product.

The processing levels are discussed in more detail in the 'dictionaries' section of the product schema discussion.

Note: Wolfgang to supply list of rules on how processing levels may be assigned.

9.7 File Format

The user making the order can request from one of a number of different delivery file formats. Currently these formats are defined as:

Id	Name
1	GeoTiff
2	ECW - ERMapper Compressed Wavelet
3	JP2 - JPEG 2000
4	ESRI -ShapeFile (Vector products only)

9.8 Packaging

By default products should be delivered inside of standard SAC packages. The packages will be placed in an ephemeral place in the file system (one where files older than 7 days are flushed daily).

9.9 Staff Order Notifications

The system can send notifications emails to staff based on product class or sensor ordered. If you look in the admin ui there is already a way to allocated users to sensors - we just need to extend that idea to include product classes as everything is not sensor based anymore.

Product types are not aggregated. From a user perspective, staff log in and choose which sensors and product classes they subscribe to by picking one or more items from a select list.

—

In product id search, the user be able to put in ranges or comma delimited lists of rows and paths too e.g.

Row: 80 Path 80

or Row: 80-83 Path: 80-83

or Row: 80,90,112 Path: 90, 101

10 Tasking Requests

The TaskingRequest model is a subclass of the Order model. Tasking requests differ from orders in that they have no products associated with them. Instead a tasking request has a sensor associated with it and other information germaine to tasking a satellite to capture and image (or other remotely sensed data) of a specific location.

10.1 Taskable Sensors

The SensorType model includes an is_taskable member which is used to determine which sensors a user may select from when making a tasking request.

11 Search Schema

This section covers the logic and schema related to search. The search process begins when the user clicks 'Search' on the main application toolbar. They have the option then of performing a simple search or an advanced search. Regardless of which search type they use, the outcome is the same - the catalogue application internally creates a collection of unserialised **SearchRecord** objects. These objects are simple mappings between a search and the products that match the criteria of that search.

The matching search records are shown to the user as a paginated table of entries, each entry corresponding to one product. If a user chooses to add that product to their basket, the SearchRecord is then serialised (saved persistently in the database). By definition, the basket can be defined as //"all serialised **SearchRecords** for a given user which have **no** Order associated with them"//. In the ordering section we describe the order process in more detail.

12 Metadata

Metadata created for products in the SAC Catalogue adheres to the ISO19115 specification. The aforementioned specification defines many different attributes that can be used to describe a product, which can result in somewhat complicated metadata documents with the majority of attributes unpopulated do to lack of sufficient information.

The standard also specifies a core set of attributes that should be present for any given metadata document.

12.1 Mandatory Core Items

These are listed in the table that follows:

1. **Dataset title (M)**
   - This is the SAC product identifier for this dataset
   - MD_Metadata > MD_DataIdentification.citation > MD_Metadata > CI_Citation.title
   - **Note:** Wolfgang in DIMS packages the ID is e.g. SPOT5.HRG.L1A but we should rather use SAC product ID here I think.
2. **Dataset reference date (M)**
   - Date on which product was acquired or created. For imagery products it should refer to the acquisition data where feasible.
   - MD_Metadata > MD_DataIdentification.citation > CI_Citation.date

3. **Dataset topic category (M)** -
    - MD_Metadata > MD_DataIdentification.topicCategory
4. **Abstract describing the dataset (M)** -
    - MD_Metadata > MD_DataIdentification.abstract
5. **Dataset language (M)**
    - This will be set to English always for any product. There is currently no provision for storing language of products in the catalogue data models.
    - MD_Metadata > MD_DataIdentification.language
6. **Metadata point of contact (M)** -
    - MD_Metadata.contact > CI_ResponsibleParty
7. **Metadata date stamp (M)**
    - This maps to the GenericProduct::product_date field
    - MD_Metadata.dateStamp

Above listing taken from section 6.5 'Core metadata for geographic subsets' of the ISO 19115:2003 specification. Items marked (M) are mandatory, (O) Optional and (C) Conditionally required.

12.2 Optional Core Items

1. **Spatial representation type (O)**
    - A geographic dataset should be represented as one of the following type codes:
        - vector
        - grid
        - textTable
        - tin
        - stereoModel
        - video
    - MD_DataIdentification.spatialRepresentationType
2. **Reference system (O)**
    - The standard allows the reference system to be described either by means of provision of projection, ellipsoid, datum, or by means of an identifier. In all cases we will use the MD_ReferenceSystem.referenceSystemIdentifier rather which should be specified in the form of an EPSG code.
    - MD_Metadata > MD_ReferenceSystem

3. **Dataset responsible party (O)**
   - This is the person / institution responsible for the metadata information.
   - MD_Metadata > MD_DataIdentification.pointOfContact > CI_ResponsibleParty
4. **Lineage (O)**
   - Lineage describes how the product was created and where if comes from. Two options for describing lineage exist (which can be used together too):
     a) Using LI_Source (with properties: description(O), scaleDenominator(O), sourceReferenceSystem(O), sourceCitation(O), sourceExtent(O))
     b) Using LI_ProcessStep (with properties: description(C), rationale(O), dataTime(O), processor(O)) where processor represents a contact person
   - MD_Metadata > DQ_DataQuality.lineage > LI_Lineage
5. **On-line resource (O)**
   - A permalink to an online record for this product. Any product will be accessible by visting a specific url based on its product ID e.g. "http://catalogue.sac.co.za/showProduct/_M-_M-_CAM2_0121_00_0404_00_061009_083518_L2A-_UTM34S/"
   - MD_Metadata > MD_Distribution > MD_DigitalTransferOption.onLine > CI_OnlineResource MD_DataIdentification.characterSet
6. **Spatial resolution of the dataset (O)** -
   - MD_Metadata > MD_DataIdentification.spatialResolution > MD_Resolution.equivalentScale or MD_Resolution.distance
7. **Distribution format (O)** -
   - MD_Metadata > MD_Distribution > MD_Format.name and MD_Format.version
8. **Additional extent information for the dataset (vertical and temporal) (O)** -
   - MD_Metadata > MD_DataIdentification.extent > EX_Extent > EX_TemporalExtent or EX_VerticalExtent
9. **Metadata file identifier (O)** -
   - (MD_Metadata.fileIdentifier)
10. **Metadata standard name (O)** -
    - MD_Metadata.metadataStandardName
11. **Metadata standard version (O)** -
    - MD_Metadata.metadataStandardVersion

Above listing taken from section 6.5 'Core metadata for geographic subsets' of the ISO 19115:2003 specification. Items marked (M) are mandatory, (O) Optional and (C) Conditionally required.

12.3 Conditional Core Items

1. **Geographic location of the dataset (C) by four coordinates or by geographic identifier**
   - The footprint of the dataset in GML taken from GenericProduct::spatial_coverage
   - If this element is not present and we are examining a product package (as opposed to a metadata only package), geographic location will be retrieved directly from teh project itself, using gdal.
   - MD_Metadata > MD_DataIdentification.extent > EX_Extent > EX_GeographicExtent > EX_BoundingPolygon or EX_GeographicBoundingBox or EX_GeographicDescription
   - **Note:** Wolfgang - DIMS should add the image footprint to the metadata, this can be achieved using EX_BoundingPolygon.
2. **Dataset character set (C)** - MD_Metadata > characterSet -
3. **Metadata language (C)**
   - This will be set to English always for any product. There is currently no provision for storing language of products in the catalogue data models.
   - MD_Metadata.language
4. **Metadata character set (C)** -
   - MD_Metadata.characterSet

Above listing taken from section 6.5 'Core metadata for geographic subsets' of the ISO 19115:2003 specification. Items marked (M) are mandatory, (O) Optional and (C) Conditionally required.

12.4 Schema Representation in XML

Taking just these core elements we can realise a minimalist document structure for raster data as listed below):

```
<?xml version="1.0" encoding="UTF-8"?>
<gmd:MD_Metadata xsi:schemaLocation="http://www.isotc211.org/2005/gmd
     http://schemas.opengis.net/iso/19139/20060504/gmd/gmd.xsd"
     xmlns:gmd="http://www.isotc211.org/2005/gmd"
     xmlns:gco="http://www.isotc211.org/2005/gco"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:gml="http://www.opengis.net/gml"
     xmlns:xlink="http://www.w3.org/1999/xlink">
  <gmd:fileIdentifier>
    <gco:CharacterString></gco:CharacterString>
  </gmd:fileIdentifier>
  <gmd:language>
    <gmd:LanguageCode codeList="http://standards.iso.org/ittf/
    PubliclyAvailableStandards/ISO_19139_Schemas/resources/
    Codelist/ML_gmxCodelists.xml#LanguageCode"
    codeListValue="eng">eng</gmd:LanguageCode>
  </gmd:language>
  <gmd:characterSet>
    <gmd:MD_CharacterSetCode codeSpace="ISOTC211/19115"
    codeListValue="MD_CharacterSetCode_utf8"
    codeList="http://www.isotc211.org/2005/resources/
    Codelist/gmxCodelists.xml#MD_CharacterSetCode">
    MD_CharacterSetCode_utf8</gmd:MD_CharacterSetCode>
```

```
</gmd:characterSet>
<gmd:hierarchyLevel>
  <gmd:MD_ScopeCode codeList="http://standards.iso.org/ittf/PubliclyAvailableStandards/
  ISO_19139_Schemas/resources/Codelist/ML_gmxCodelists.xml#
  MD_ScopeCode" codeListValue="dataset">dataset</gmd:MD_ScopeCode>
</gmd:hierarchyLevel>
<gmd:contact>
  <gmd:CI_ResponsibleParty>
    <gmd:organisationName>
      <gco:CharacterString>Test org</gco:CharacterString>
    </gmd:organisationName>
    <gmd:contactInfo>
      <gmd:CI_Contact>
        <gmd:address>
          <gmd:CI_Address>
            <gmd:electronicMailAddress>
              <gco:CharacterString>test@test.com</gco:CharacterString>
            </gmd:electronicMailAddress>
          </gmd:CI_Address>
        </gmd:address>
      </gmd:CI_Contact>
    </gmd:contactInfo>
    <gmd:role>
      <gmd:CI_RoleCode codeList="http://standards.iso.org/ittf/
         PubliclyAvailableStandards/ISO_19139_Schemas/resources/
         Codelist/ML_gmxCodelists.xml#CI_RoleCode" codeListValue="pointOfContact">
         pointOfContact</gmd:CI_RoleCode>
    </gmd:role>
  </gmd:CI_ResponsibleParty>
</gmd:contact>
<gmd:dateStamp>
  <gco:Date>2011-03-01</gco:Date>
</gmd:dateStamp>
<gmd:metadataStandardName>
  <gco:CharacterString>ISO19115</gco:CharacterString>
</gmd:metadataStandardName>
<gmd:metadataStandardVersion>
  <gco:CharacterString>2003/Cor.1:2006</gco:CharacterString>
</gmd:metadataStandardVersion>
<gmd:identificationInfo>
  <gmd:MD_DataIdentification>
    <gmd:citation>
      <gmd:CI_Citation>
        <gmd:title>
          <gco:CharacterString>Test title</gco:CharacterString>
        </gmd:title>
        <gmd:date>
          <gmd:CI_Date>
            <gmd:date>
              <gco:Date>2011-03-01</gco:Date>
            </gmd:date>
            <gmd:dateType>
              <gmd:CI_DateTypeCode codeList="http://standards.iso.org/ittf/
              PubliclyAvailableStandards/ISO_19139_Schemas/resources/Codelist/
              ML_gmxCodelists.xml#CI_DateTypeCode"
              codeListValue="creation">creation</gmd:CI_DateTypeCode>
            </gmd:dateType>
          </gmd:CI_Date>
        </gmd:date>
        <gmd:identifier>
          <gmd:RS_Identifier>
            <gmd:code>
              <gco:CharacterString>1</gco:CharacterString>
            </gmd:code>
            <gmd:codeSpace>
              <gco:CharacterString>1</gco:CharacterString>
            </gmd:codeSpace>
          </gmd:RS_Identifier>
        </gmd:identifier>
      </gmd:CI_Citation>
    </gmd:citation>
    <gmd:abstract>
      <gco:CharacterString>Test abstract</gco:CharacterString>
    </gmd:abstract>
    <gmd:topicCategory>
      <gmd:MD_TopicCategoryCode>environment</gmd:MD_TopicCategoryCode>
    </gmd:topicCategory>
    <!-- TO BE ADDED
      <igmd:extent>
        <gmd:EX_Extent>
          <gmd:geographicElement>
            <gmd:EX_BoundingPolygon>
              <gmd:polygon>
                <gml:outerBoundaryIs>
```

```
                    <gml:LinearRing>
      -->
    </gmd:MD_DataIdentification>
  </gmd:identificationInfo>
  <gmd:dataQualityInfo>
    <gmd:DQ_DataQuality>
      <gmd:scope>
        <gmd:DQ_Scope>
          <gmd:level>
            <gmd:MD_ScopeCode codeListValue="dataset"
            codeList="http://standards.iso.org/ittf/PubliclyAvailableStandards/
            ISO_19139_Schemas/resources/Codelist/ML_gmxCodelists.xml#
            MD_ScopeCode">dataset</gmd:MD_ScopeCode>
          </gmd:level>
        </gmd:DQ_Scope>
      </gmd:scope>
    </gmd:DQ_DataQuality>
  </gmd:dataQualityInfo>
</gmd:MD_Metadata>
```

12.5 Editing Metadata

There are a few different potential sources of metadata:

- Online editors such as the http://www.inspire-geoportal.eu/EUOSME/ editor
- DIMS generated packages containing metadata
- The QGIS metadata preparation tool being built to complement the online catalogue
- This catalogue will generate metadata for products when their metadata records are being exported (e.g. when a user wishes to save their search results as a metadata record, when their cart product descriptions are downloaded or when products themselves are downloaded).

12.6 Required modifications to the ISOMetadata.xml

This list is about the modifications that has to be done to the ISOMetadata.xml in order to ingest the DIMS packages.

1. We do need the spatial coverage extent for the image footprint, so the extent must be specified with EX_BoundingPolygon instead of EX_GeographicBoundingBox

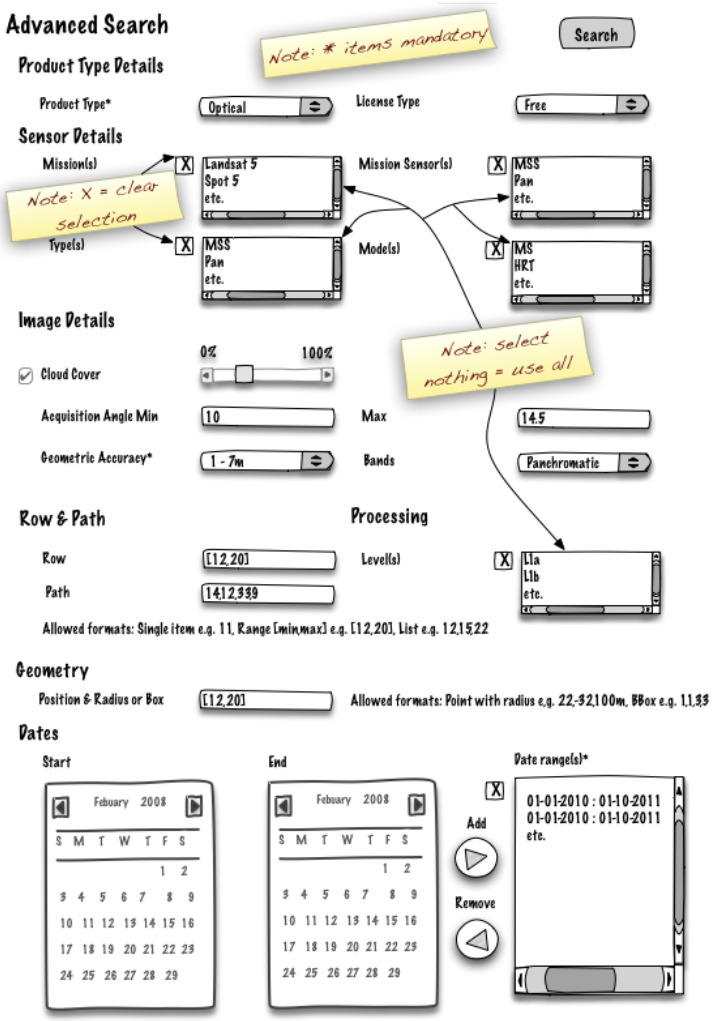
-

13 Searching for data on the Catalogue

13.1 Basic Search

13.2 Advanced Search

13.2.1 Optical Products



An advanced search for optical products allows the user to create a refined subset of OpticalProduct records based on various properties. The options selected in the advanced search form are cumulative.

In the wireframe above, you can see the options available in advanced search.

Sensor Details

The sensor details area of the form allows the user to define which products should be found based on the sensors that acquired them.

- When selecting one or more missions, the sensors, modes, types and bands entries will be filtered to show only those items permissible for the given missions.
- When selecting one or more sensors, the modes, types and bands entries will be filtered to show only those items permissible for the given sensors.
- When selecting one or more types, the mode and band entries will be filtered to show only those items permissible for the given types.
- When selecting one or more modes, the band entries will be filtered to show only those items permissible for the given modes.

Image Details

The image details area of the form allows the user to define which products should be found based on the product / image properties.

Cloud cover can optionally be used to filter products that have been determined to have less than a given amount of cloud cover. Images with NULL cloud_cover will be assumed to have 100% cloud cover for purposes of filtering. If the checkbox next to the cloud cover slider is unchecked, cloud cover will be ignored when filtering.

The **acquisition angle** is used to limit search results to products that were acquired with the sensor orientated between a range of viewing angles. Images with NULL sensor_viewing_angle will be assumed to be excluded when filtering by sensor viewing angle. If either the min or max value is unpopulated, this will be ignored in the search.

Geometric accuracy is used to filter products based on image pixel size. On a technical note, the pixel size is determined by checking the GenericImageryProduct::geometric_resolution (which is an inherited property of optical product). The reason this is used rather than the AcquisitionMode::geometric_resolution is that the product may have been resampled during processing so (with the exception of level 1A products), the resolution may not correspond to the acquisition mode any more. For ease of use, geometric accuracy is grouped as shown in the table below, and the user may select a single entry from this list.

Class	Value Range
< 1m	0 - 1000mm
1m - 7m	1001 - 7000mm
7m - 25m	7001 - 25000mm
25m - 70m	25001 - 70000mm
70m - 1km	70001 - 1000000mm
> 1km	1000000mm +

Note: Internally, geometric resolution is specified in mm

Wolfgang: mm or m?

Bands is used to identify the spectral resolution of the products being searched for. A user friendly grouping is listed in a select box allowing the user to select a single band range. These are:

Name	Notes
Panchromatic	Single band imagery
Truecolour RGB (3)	Three bands representing red green blue
Multispectral (4-8)	4 to 8 bands
Superspectral (9-40)	9 to 40 bands
Hyperspectral (>40)	more than 40 bands

Technical note: As with geometric accuracy, the number of bands used for filtering is taken from GenericImageryProduct::band_count rather than AcquisitionMode::band_count.

Row / Path Ranges

The specification of row and path ranges can be one in one of three ways:

1. A range e.g. [0,15] means include all rows/paths inclusively from 0 to 15.
2. A list e.g. 1,5,3 means include specifically row or paths numbered 1, 5 and 3.
3. A single item e.g. 10 means include only that row/path

The sematics of what constitutes a row and path (in a geospatial sense) varies from sensor to sensor. The search algorithm makes no special allowance for this so it should be noted that a row/path search of 30,50 for two different sensors my return imagery covering disparate areas.

Processing levels

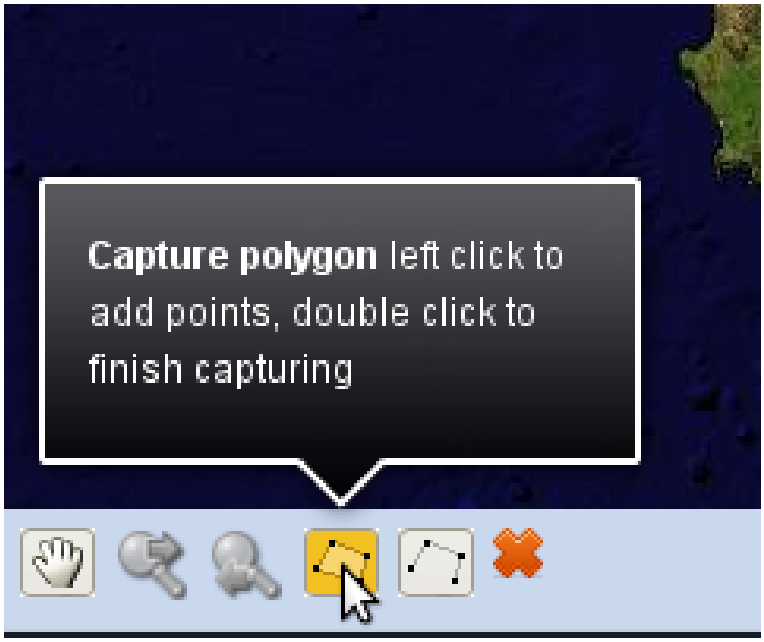
The products can be filtered according to their processing level. The list of processing level list is not dynamic (i.e. it does not ajust its entries based on mission, sensor etc. selections) as it would be prohibitively CPU intensive to do so. As such there is no guarantee of being at least one product for each of the listed processing levels.

One or more processing levels can be selected. If none are selected, it is assumed that products at all levels should be queried when searching.

Geometry

A geometry can be defined for a search in one of four ways:

1. Digitising an area on the map directly
2. Uploading a shapefile or kml demarcating the area of interest
3. Specifying point and radius geometry in an input box
4. Specifying a bounding box in an input box



For digitising, the user should select the 'capture polygon' tool on the map.

For the geometry upload functions, only the first feature in the uploaded shapefile / kml will be used. The shapefile should have a polygon geometry type and have a Coordinate Reference System of EPSG:4326 (Geographic WGS84).

The input box for geometry can be used to create a circular geometry by entering the coordinate of the center point and a radius in kilometers. For example:

```
20.5,-32.3,100
```

The values should be entered as easting (use negative number to indicate west), northing (use negative number to indicate south), radius (in km). The easting/northing values are specified in decimal degrees.

Finally, the bounding box can also be entered in the form:

```
xmin,ymin,xmax,ymax
```

For example:

```
20,-34,22,-32
```

Date Ranges

When conducting a search with the advanced search tool, the user can specify one or more date ranges. The start date for any date range will be restricted to be no earlier **1 January 1972**. The end date will be restricted to be no later than the current date. No facility is made for including time in the date range.

The process of defining date ranges consists of:

- Selecting a start date in the 'start' calendar

- Selecting an end date in the 'end' calendar
- Clicking the 'add' icon to add the date range to the range list

The user can remove a range from the list by selecting it and clicking the 'remove' icon.

Executing the search

Once the user has completed the search definition process, pressing the 'search' button will cause the search request to be posted to the server. If there are any validation errors, the user will be returned to the search form and the validation errors will be indicated. Assuming there were no errors, the user is redirected to the search results page.

14 Informix SPOT Catalogue Notes

This section is broken up into the following parts:

1. a description of the ACS port and data migration process
2. technical information on how to connect to informix from python
3. miscellaneous tips and tips relating to using the informix database

14.1 Overview of the data migration process

The process of data migration seeks to largely clone the informix ACS catalogue into a postgresql database, and then use that as the basis of running various import steps in order to migrate key parts of the ACS database into the SAC Catalogue.

There are two things to consider here:

1. migration of metadata
2. migration of product thumbnails

Metadata records are defined in a complex of different tables which need to be queried in order to be able to obtain all the data related to a specific product. The informix acs catalogue stores all thumbnails as whole segments within blobs inside the database and these need to be extracted, georeferenced and clipped into individual scenes.

14.2 Technical notes for informix access via python

This section covers the installation and setup process for the informix python client so that you can connect to the server from a separate linux box using python.

Download the InformixDB driver for python from:

`http://sourceforge.net/project/showfiles.php?group_id=136134`

And the Informix client sdk from:

```
http://www14.software.ibm.com/webapp/download/preconfig.jsp?
    id=2007-04-19+14%3A08%3A41.173257R&S_TACT=104CBW71&S_CMP=
```

(write the above url on a single line)

If the above link doesnt work for you (it seems to contain a session id), go to the

```
http://www14.software.ibm.com
```

website and search for

```
3.50.UC4
```

using the search box near the top right of the page. Downloading requires an IBM id etc. which you can sign up for if you dont have one.

Note: You will need to get the appropriate download for your processor type. For Lion, which is running ubuntu server x86_64, I downloaded the sdb bundle called:

```
IBM Informix Client SDK V3.50.FC4 for Linux (x86) RHEL 4, 64bit
clientsdk.3.50.FC4DE.LINUX.tar  (72MB)
```

Note 2: Even though it says Red Hat Enterprise Edition (RHEL) you can use it on ubuntu servers too.

After you have downloaded the client sdk do the following to install (below is a log of my install process).

```
sudo adduser informix
Adding user ‘informix’ ...
Adding new group ‘informix’ (1003) ...
Adding new user ‘informix’ (1003) with group ‘informix’ ...
Creating home directory ‘/home/informix’ ...
Copying files from ‘/etc/skel’ ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for informix
Enter the new value, or press ENTER for the default
Full Name []: Informix
Room Number []:
Work Phone []:
Home Phone []:
Other []:
Is the information correct? [Y/n] Y
[linfiniti:timlinux:DownloadDirector] sudo ./installclientsdk




         Initializing InstallShield Wizard........
          Launching InstallShield Wizard........


-------------------------------------------------------------------------------
Welcome to the InstallShield Wizard for IBM Informix Client-SDK Version 3.50

The InstallShield Wizard will install IBM Informix Client-SDK Version 3.50 on
your computer.
To continue, choose Next.

IBM Informix Client-SDK Version 3.50
IBM Corporation
http://www.ibm.com
```

```
Press 1 for Next, 3 to Cancel or 4 to Redisplay [1] 1

-------------------------------------------------------------------------------
     International License Agreement for Non-Warranted Programs

     Part 1 - General Terms

     BY DOWNLOADING, INSTALLING, COPYING, ACCESSING, OR USING THE PROGRAM
     YOU AGREE TO THE TERMS OF THIS AGREEMENT. IF YOU ARE ACCEPTING THESE
      TERMS ON BEHALF OF ANOTHER PERSON OR A COMPANY OR OTHER LEGAL
      ENTITY, YOU REPRESENT AND WARRANT THAT YOU HAVE FULL AUTHORITY TO
      BIND THAT PERSON, COMPANY, OR LEGAL ENTITY TO THESE TERMS. IF YOU DO
      NOT AGREE TO THESE TERMS,



- DO NOT DOWNLOAD, INSTALL, COPY, ACCESS, OR USE THE PROGRAM; AND



- PROMPTLY RETURN THE PROGRAM AND PROOF OF ENTITLEMENT TO THE PARTY

Press Enter to continue viewing the license agreement, or, Enter "1" to accept
the agreement, "2" to decline it or "99" to go back to the previous screen, "3"
 Print.

1

Press 1 for Next, 2 for Previous, 3 to Cancel or 4 to Redisplay [1] 1

-------------------------------------------------------------------------------
IBM Informix Client-SDK Version 3.50 Install Location

Please specify a directory or press Enter to accept the default directory.

Directory Name: [/opt/IBM/informix] /usr/informix

Press 1 for Next, 2 for Previous, 3 to Cancel or 4 to Redisplay [1] 1

-------------------------------------------------------------------------------
Choose the setup type that best suits your needs.

[X] 1 - Typical
        The program will be installed with the suggested configuration.
        Recommended for most users.

[ ] 2 - Custom
        The program will be installed with the features you choose.
        Recommended for advanced users.

To select an item enter its number, or 0 when you are finished: [0]


Press 1 for Next, 2 for Previous, 3 to Cancel or 4 to Redisplay [1] 1

-------------------------------------------------------------------------------
IBM Informix Client-SDK Version 3.50 will be installed in the following
location:

/usr/informix

with the following features:

Client
Messages
Global Language Support (GLS)

for a total size:

 91.8 MB

Press 1 for Next, 2 for Previous, 3 to Cancel or 4 to Redisplay [1] 1

Installing IBM Informix Client-SDK Version 3.50. Please wait...


|-----------|-----------|-----------|-----------|
0%         25%         50%         75%        100%
||||||||||||||||||||||||||||||||||||||||||||||||||

Creating uninstaller...
Performing GSKit installation for Linux ...
```

```
Branding Files ...
Installing directory .
Installing directory etc
Installing directory bin
Installing directory lib
Installing directory lib/client
Installing directory lib/client/csm
Installing directory lib/esql
Installing directory lib/dmi
Installing directory lib/c++
Installing directory lib/cli
Installing directory release
Installing directory release/en_us
Installing directory release/en_us/0333
Installing directory incl
Installing directory incl/esql
Installing directory incl/dmi
Installing directory incl/c++
Installing directory incl/cli
Installing directory demo
Installing directory demo/esqlc
Installing directory demo/c++
Installing directory demo/cli
Installing directory doc
Installing directory doc/gls_api
Installing directory doc/gls_api/en_us
Installing directory doc/gls_api/en_us/0333
Installing directory tmp
Installing directory gsk
Installing directory gsk/client
Installing directory gskit
Installing directory gsk
Installing directory gsk/client

IBM Informix Product:       IBM INFORMIX-Client SDK
Installation Directory: /usr/informix

Performing root portion of installation of IBM INFORMIX-Client SDK...


Installation of IBM INFORMIX-Client SDK complete.

Installing directory etc
Installing directory gls
Installing directory gls/cm3
Installing directory gls/cv9
Installing directory gls/dll
Installing directory gls/etc
Installing directory gls/lc11
Installing directory gls/lc11/cs_cz
Installing directory gls/lc11/da_dk
Installing directory gls/lc11/de_at
Installing directory gls/lc11/de_ch
Installing directory gls/lc11/de_de
Installing directory gls/lc11/en_au
Installing directory gls/lc11/en_gb
Installing directory gls/lc11/en_us
Installing directory gls/lc11/es_es
Installing directory gls/lc11/fi_fi
Installing directory gls/lc11/fr_be
Installing directory gls/lc11/fr_ca
Installing directory gls/lc11/fr_ch
Installing directory gls/lc11/fr_fr
Installing directory gls/lc11/is_is
Installing directory gls/lc11/it_it
Installing directory gls/lc11/ja_jp
Installing directory gls/lc11/ko_kr
Installing directory gls/lc11/nl_be
Installing directory gls/lc11/nl_nl
Installing directory gls/lc11/no_no
Installing directory gls/lc11/os
Installing directory gls/lc11/pl_pl
Installing directory gls/lc11/pt_br
Installing directory gls/lc11/pt_pt
Installing directory gls/lc11/ru_ru
Installing directory gls/lc11/sk_sk
Installing directory gls/lc11/sv_se
Installing directory gls/lc11/th_th
Installing directory gls/lc11/zh_cn
Installing directory gls/lc11/zh_tw

IBM Informix Product:       Gls
Installation Directory: /usr/informix
```

```
Performing root portion of installation of Gls...


Installation of Gls complete.

Installing directory etc
Installing directory msg
Installing directory msg/en_us
Installing directory msg/en_us/0333

IBM Informix Product:       messages
Installation Directory: /usr/informix

Performing root portion of installation of messages...


Installation of messages complete.


-------------------------------------------------------------------------------
The InstallShield Wizard has successfully installed IBM Informix Client-SDK
Version 3.50. Choose Finish to exit the wizard.

Press 3 to Finish or 4 to Redisplay [3]
```

Note that trying to install it to another directory other than /usr/informix will cause the db adapter build to fail (and various other issues). So dont accept the default of /opt/IBM/informix and rather use /usr/informix

Now build the python informix db adapter:

```
cd /tmp/InformixDB-2.5
python setup.py build_ext
sudo python setup.py install
```

Now ensure the informix libs are in your lib search path:

```
sudo vim /etc/ld.so.conf
```

And add the following line:

```
/usr/informix/lib/
/usr/informix/lib/esql
```

Then do

```
sudo ldconfig
```

14.2.1 Making a simple python test

First you need to add a line to informix's sqlhosts file:

```
sudo vim /usr/informix/etc/sqlhosts
```

And add a line that looks like this:

```
#catalog2 added by Tim
#name, protocol, ip, port
catalog2        onsoctcp        196.35.94.210  1537
```

Next you need to export the INFORMIXSERVER environment var:

```
export INFORMIXSERVER=catalog2
```

I found out that it is running on port 1537 by consulting the /etc/services file on the informix server. Now lets try our test connection. This little script will make a quick test connection so you can see if its working:

```
#!/usr/bin/python

import sys
import informixdb  # import the InformixDB module

# -------------------------------------
# open connection to database 'stores'
# -------------------------------------
conn = informixdb.connect('catalogue@catalog2', user='informix', password='')

# ---------------------------------
# allocate cursor and execute select
# ---------------------------------
cursor1 = conn.cursor(rowformat = informixdb.ROW_AS_DICT)
cursor1.execute('select * from t_file_types')

for row in cursor1:

  # -----------------------------------------
  # delete row if column 'code' begins with 'C'
  # -----------------------------------------
  print "%s %s" % (row['id'], row['file_type_name'])
# -------------------------------------
# commit transaction and close connection
# -------------------------------------
conn.close()

sys.exit(0);
```

Note that the documentation for the python InformixDB module is available here:

```
http://informixdb.sourceforge.net/manual.html
```

And the documentation for the Informix SQL implementation is here:

```
http://publib.boulder.ibm.com/infocenter/idshelp/v10
```

14.3 Trouble shooting and general tips

14.3.1 WKT representation of GeoObjects

Informix uses its own representation of geometry objects. There are two extensions for informix that deal with spatial data : Geodetic and Spatial. It seems we have only geodetic extension at SAC and thus can't use ST_foo functions to work with geometry fields. For Geodetic we need to alter a value in the GeoParam table in order to change what formats are output / input. From the manual:

```
Converting Geodetic to/from OpenGIS Formats

Geodetic does not use functions to convert data to a specific format.

Instead, the GeoParam metadata table manages the data format for transmitting
data between client and server. If the "data format" parameter is set to "OGC",
then binary i/o is in WKB format and text i/o is in WKT format. (For specific
details, see Chapter 7 in the Informix Geodetic DataBlade Module User's Guide).
```

You can override the representation type that should be returned so that you get e.g. WKT back instead. Consider this example:

```
-- set output format to 3
update GeoParam set value = 4 where id =3;
-- show what the format is set to now
select * from GeoParam where id = 3;
-- display a simple polygon
select first 1 geo_time_info from t_localization;
-- revert it to informix representation
update GeoParam set value = 0 where id =3;
-- display the polygon back in native informix representation
select first 1 geo_time_info from t_localization;
--verify that the format is reverted correctly
select * from GeoParam where id = 3;
```

Which produces output like this:

```
id       3
name     data format
value    4
remarks  This parameter controls the external text & binary format of GeoObjects.
         It is not documented in the 3.0 version of the user's guide.  See release
         notes for more info.



geo_time_info  POLYGON((28.73 -15.35, 28.969999 -13.79, 27.34 -13.55, 27.1 -15.
                        11, 28.73 -15.35))



geo_time_info  GeoPolygon((((-15.35,28.73),(-13.79,28.969999),(-13.55,27.34),
               (-15.11,27.1))),ANY,(1987-04-26 07:34:45.639,1987-04-26
               07:34:45.639))



id       3
name     data format
value    0
remarks  This parameter controls the external text & binary format of GeoObject
         s.  It is not documented in the 3.0 version of the user's guide.  See
         release notes for more info.
```

14.3.2 When things go wrong on the informix server

Record Lock Issues

If the client does not cleanly disconnect it can leave records locked. You may see a message like this from dbaccess when trying to do an interactive query:

```
244: Could not do a physical-order read to fetch next row.
107: ISAM error:  record is locked.
```

There are probably solutions that are better than this, but the most robust way of dealing with the issue is to restart the informix database:

```
ssh informix@informix
cd /home/informix/bin
onmode -k
```

You will then get prompted like this:

```
This will take Informix Dynamic Server 2000 OFF-LINE -
Do you wish to continue (y/n)? y

There are 1 user threads that will be killed.
Do you wish to continue (y/n)? y
```

Afterwards, you can bring up the database like this:

```
oninit
```

The record locks should have been cleared at this point.

DBAccess Unresponsive

Collect diagnostics:
[101] catalog2:/home/informix> onstat -V Informix Dynamic Server 2000 Version 9.21.UC4 Software Serial Number AAD#J130440
[101] catalog2:/home/informix> onstat -a

Nightly Informix Compact Job

```
ssh informix@informix
crontab -l
[101] catalog2:/home/informix> crontab -l
no crontab for informix
```

So now we make a little bash script:

```
#!/bin/bash

# A simple bash script to  be invoked by CRON on a nightly basis
# To enable add to your crontab like so:
#
# ------------------------------------------------------------
#
# Run informix stats update nightly to keep responsiveness good
# Job will run 5 min after midnight
# 5 0 * * * /home/informix/nightly_cron.sh
#
# ------------------------------------------------------------
#
# Actual script follows:
#
# Tim Sutton, May 2009
#

# Update the db stats on a nightly basis:

date >> /tmp/informix_stats_update_cron_log.txt
echo "Nightly stats update running" >> \
       /tmp/informix_stats_update_cron_log.txt
echo "update statistics high;" | \
       dbaccess catalogue >> /tmp/informix_stats_update_cron_log.txt
```

And then set up a nightly cronjob to run it:

```
crontab -e
```

Now add this:

```
# Added by Tim for others to see how crontab works
#*     *     *     *     *  command to be executed
#-     -     -     -     -
#|     |     |     |     |
#|     |     |     |     +----- day of week (0 - 6) (Sunday=0)
#|     |     |     +------- month (1 - 12)
#|     |     +--------- day of month (1 - 31)
#|     +----------- hour (0 - 23)
#+------------- min (0 - 59)


# Run a test command every minute to see if crontab is working nicely
# comment out when done testing
#*/1 * * * * date >> /tmp/date.txt

# Run informix stats update nightly to keep responsiveness good
# Job will run 5 min after midnight
5 0 * * * /home/informix/nightly_cron.sh
```

14.3.3 File System

```
root@informix's password:
Last login: Tue Sep  9 12:57:53 2008 from :0
[root@catalog2 /root]# mount
/dev/sda6 on / type ext2 (rw)
none on /proc type proc (rw)
usbdevfs on /proc/bus/usb type usbdevfs (rw)
/dev/sda2 on /boot type ext2 (rw)
/dev/sda10 on /home type ext2 (rw)
/dev/sda8 on /tmp type ext2 (rw)
/dev/sda5 on /usr type ext2 (rw)
/dev/sda9 on /var type ext2 (rw)
none on /dev/pts type devpts (rw,gid=5,mode=620)
/dev/sdb1 on /mnt/disk1 type ext2 (rw)
/dev/sdc1 on /mnt/disk2 type ext2 (rw)
automount(pid458) on /misc type autofs (rw,fd=5,pgrp=458,minproto=2,maxproto=3)
```

14.3.4 Schema dump of informix databases

Its useful to be able to see the schema of databases so you can understand how it was put together. The following command will dump the catalogue2 (SAC Production database) schema to a text file. **Note:** No data is dumped in this process.

```
dbschema -t all -d catalogue catalogue_schema.sql
```

14.3.5 Listing system and user functions

To see what functions are installed in the database do:

```
select procname from sysprocedures;
```

To see full details of a function:

```
select * from sysprocedures where procname="lotofile";
```

14.3.6 Problems running functions

If you try to run a function that you know exists, but you get an error message like this:

```
_informixdb.DatabaseError: SQLCODE -674 in PREPARE:
IX000: Routine (lotofile) can not be resolved.
```

It probably means you passed the incorrect number or type of parameters to the function.

14.3.7 Accessing the server Interactively

```
ssh 196.35.94.210 -l informix
```

Interactive database access:

```
dbaccess
```

14.4 Command line batch processing

Add some sql commands to a text file:

```
vim /tmp/tim.sql
```

Some commands:

```
select geo_time_info from ers_view;
```

Save and run, redirecting output to another text file:

```
dbaccess catalogue < /tmp/tim.sql >> /tmp/tim.out
```

14.4.1 Command line processing using echo

Handy for quickly running once off commands of from bash scripts.

```
echo "select * from t_file_types" | dbaccess catalogue
```

14.4.2 Changing geotype to wkt

For batch export to the django catalogue the geometries need to be exported as wkt (well known text) which is not the type used internally for the spot catalogue.

```
echo "update GeoParam set value = 0 where id =3;" | dbaccess catalogue
```

14.4.3 Reverting geotype to informix format

To set geometry output back informix representation and restoring normal catalogue functioning do:

```
echo "update GeoParam set value = 4 where id =3;" | dbaccess catalogue
```

14.4.4 Informix environment preparation

```
export INFORMIXSERVER=catalog2
```

from the shell to make sure you have the informix env set up

```
Superclasses - OK 3 Records
DataMode - OK 3 Records
EllipsoidType - OK 2 Records
ErsCompMode - OK 2 Records
FileType - OK 18 Records
HeaderType - OK 7 Records
Satellte - OK 9 Records
Satellite - OK 15 Records
SpotAcquisitionMode - OK 3 Records
Station - OK 110 Records
Medium - OK: 157277 Records, Failed:0 Records.
Localization - OK: 1179257 Records, Failed:0 Records.
SegmentCommon - OK: 157277 Records, Failed:0 Records.
```

Note: Also probably no longer needed!

14.5 Backup and Restore of the postgres ACS clone

To backup the ACS postgres database do:

```
pg_dump -f acs-‘date +%a%d%b%Y‘.sql.tar.gz -x -O -F tar acs
```

To restore the postgres database do:

```
pg_restore -F t acsThu21May2009.sql.tar.gz |psql acs
```

15 Procedures for importing data from various sources into the catalogue

Note: This document should be considered compulsory reading before you attempt to import any data into the catalogue.

Note2: This document *must* be kept up to date when you make changes to import scripts etc.

The catalogue system provides search access to metadata describing acquisitions that have taken place from a variety of sensors. This metadata needs to be lodged in the database in one way or another. Different sensors have different entry points into the system - and this document tries to cover the various permutations and procedures for lodging data into the database.

15.1 Legacy ACS System

15.2 SPOT Image Data

15.3 Sumbandilasat

For sumbandilasat the procedure for import at the moment boils down to this:

- Wolfgang / other SAC staffer performs initial L1Ab processing of imagery
- Products are placed on the SAC storage array and an email is sent to Tim detailing the names of the new product directories. (Typically they will be under `/cxfs/SARMES/S/INT/RI/SS1/`
- The products are copied over to LION into `/mnt/cataloguestorage/imagery_processing/sumbandi` e.g. `rsync -ave ssh cheetah:/cxfs/SARMES/S/INT/RI/SS1/2* .`
- Before rsyncing, it would be worth noting which products are already processed e.g. “20100409-20100712 20100801_20100830 20100901_20100910 20100901_20100922 20100927_20101014 20101018_20101108 20101109_20101112 20101116_20101119“
- The .shp project file is then imported into the sac database in the import schema to the ’sumb’ table

- The scripts/sort_sumb_imagery.py script is then run. This converts the sumb pix images to tif and then files them under imagery master copies in the L1Ab folder as shown below.

```
imagery_mastercopies
+-- C2B      <-- CBERS
|   +-- 1Aa
|   +-- 1Ab
+-- S-C      <-- SACC
|   +-- 1Ab
+-- ZA2      <-- sumbandilasat
    +-- 1Aa
    +-- 1Ab
```

- The scripts/sort_sumb_raw_imagery.py script is then run. This archives the raw folder and files it into the L1Aa folder as shown above. Note: This step will be merged with the above step for convenience.

After this process the new data should be searchable in the catalogue, thumbnails should be available, and the raw products should be downloadable.

15.3.1 Copying the product folder over to LION

Currently we pull the data over from the storage array to LION. This is carried out using rsync. Here is an example of copying a DIMS project folder over:

```
cd /mnt/cataloguestorage/imagery_processing/sumbandilasat
rsync -ave ssh cheetah:/S/INT/RI/SS1/20100901_20100910 .
```

The copied over project file should have a structure something like this:

```
20100801_20100830
+-- imp
|   +-- ThNl
+-- raw
    +-- I0049
    +-- I0085
    ...etc
```

So the data in imp will be converted from pix into tif and made available as L1Ab products. The data in Thnl will be imported as prodct thumbnails or 'quicklooks'. The folders under raw will be archved using a filename that matches their sac product ID and made available as downloads.

15.3.2 Importing the report file

Once the project folder has been carried over to LION, you need to import the report file into the database. To do this the report file needs to be copied over to ELEPHANT (the database server), the temporary sumb import table cleared and the new report file brought in to populate that table.

There is a django model called 'Sumb' which maps to this temporary import table

- it is not used for anything besides data import and can be safely removed if you do not use Sumbandilasat on your catalogue deployment.

  The report file comes in two forms, a Geomatica 'PIX' file and a 'Shapefile' (which is actually a collection of a number of files).

```
SARMES_SS1_20100409-20100712_rep.dbf
SARMES_SS1_20100409-20100712_rep.pix
SARMES_SS1_20100409-20100712_rep.prj
SARMES_SS1_20100409-20100712_rep.shp
SARMES_SS1_20100409-20100712_rep.shx
```

To move these files (the name will differ by product folder so this is just an example) to elephant we do:

```
scp -P 8697 SARMES_SS1_20100409-20100712_rep.* elephant:/tmp/
```

You will need login credentials for elephant of course. Once the files are transferred, you need to log in to elephant (196.35.94.197), clear the import.sumb temporary table and import the report file:

```
ssh -p 8697 elephant
cd /tmp
```

Now open a db session and clear the sumb temporary table:

```
psql sac
delete from import.sumb;
\q
```

Now load the report shapefile into the temporary table (lines wrapped for readability):

```
shp2pgsql -a -s 4326 -S \
  /tmp/SARMES_SS1_20100409-20100712_rep.shp \
  import.sumb | psql sac
```

If you have a batch of report files to import in one go you can do it with a bash one liner like this:

```
for FILE in *.shp; do shp2pgsql -a -s 4326 -S $FILE import.sumb | psql sac; done
```

After importing, you can verify that all product records were loaded in the metadata table like this:

```
psql sac
sac=# select count(*) from import.sumb;
\q
```

Which should output something like this:

```
 count
 -------
  352
 (1 row)
```

Now log out of elephant and we will continue with the import on LION.

15.3.3 Unified product migration

Our goal here is to convert the Sumbandilasat data into the SAC Unified Product Model (UPM). The purpose of the UPM is to use a common product table for all sensor types - it includes only cross cutting attributes and does not try to model sensor specific attributes of a product. There are a few UPM specialisations - UPM-O for optical products, UPM-R for radar products and UPM-A for atmospheric products. Since Sumbandilasat is an optical product, metadata records will be lodged as UPM-O.

```
cd /opt/sac_catalogue/sac_live
```

now edit ‘`scripts/sumb_importer.py`‘ and at the bottom of the file populate the list of project folders to process. e.g.

```
def run():
  myProjectsList = [
      "20101122_20101201",
      "20101206_20101213",
      "20110125_20110214",
      "20110215_20110228",
                  ]
```

Also make sure that ‘`mSourcePath`‘ at the top of the file is correct (you would typically not need to change it).

Now run the script by typing:

```
cd <path to catalogue dir>
source ../python/bin/activate
python manage.py runscript sumb_importer
```

To achieve this we will run a python script that will do the work for us.

15.3.4 Downloadable Products

15.4 CBERS

15.5 SACC

16 Procedures for importing data from DIMS packages into the catalogue

Note: This document should be considered compulsory reading before you attempt to import any data into the catalogue.

Note2: This document *must* be kept up to date when you make changes to import scripts etc.

16.1 Importing the packages from a pickup folder

The import process is done by a Django management command that can be called by a cron job.

When a package is successfully imported it is deleted from the filesystem (a command flag can be used to avoid this behaviour).

To see all available options you can call the command with ‘-h‘ or ‘help‘ parameter:

```
$ python manage.py dims_ingest -h
Usage: manage.py dims_ingest [options]

Import into the catalogue all DIMS packages in a given folder, SPOT-5 OpticalProduct only

Options:
  -v VERBOSITY, --verbosity=VERBOSITY
                        Verbosity level; 0=minimal output, 1=normal output,
                        2=all output
  --settings=SETTINGS   The Python path to a settings module, e.g.
                        "myproject.settings.main". If this isn't provided, the
                        DJANGO_SETTINGS_MODULE environment variable will be
                        used.
  --pythonpath=PYTHONPATH
                        A directory to add to the Python path, e.g.
                        "/home/djangoprojects/myproject".
  --traceback           Print traceback on exception
  -f FOLDER, --folder=FOLDER
                        Scan the given folder, defaults to current.
  -i, --store_image     Store the original image data extracted from the
                        package.
  -g GLOB, --glob=GLOB  A shell glob pattern for files to ingest.
  -t, --test_only       Just test, nothing will be written into the DB.
  -o OWNER, --owner=OWNER
                        Name of the Institution package owner. Metadata will
                        be used if available, the program will fail if no
                        metadata are available and no name is provided.
  -s CREATING_SOFTWARE, --creating_software=CREATING_SOFTWARE
                        Name of the creating software. Defaults to: SARMES1
  -l LICENSE, --license=LICENSE
                        Name of the license. Defaults to: SAC Commercial
                        License
  -k, --keep            Do not delete the package after a successful import.
  --version             show program's version number and exit
  -h, --help            show this help message and exit
```

16.1.1 Options in detail

-f FOLDER, –folder=FOLDER This is the folder in which the packages to ingest are searched, defaults to the current working directory. Example: '-f /var/dims_packages'

-i, –store_image Store the original image data extracted from the package. This flag indicates to the program that the original image must be stored locally. The destination folder is calculated by chaining `settings.IMAGERY_ROOT` and the result from the function call `GenericSensorProduct.imagePath()`, the image is compressed with `bzip2` and added a ".bz2" suffix.

-g GLOB, –glob=GLOB A shell glob pattern for files to ingest. Defaults to "*.tar.gz". Example: `-g "*.tgz"`

-t, –test_only Just test, nothing will be written into the DB or copied to the filesystem.

-o OWNER, –owner=OWNER Name of the institution, as a string. Defaults to: None. Example: `-o "Satellite Applications Centre"` Note: the institution will be created if it does not exist. Note: the institution will be read from metadata if not specified in the options.

-s CREATING_SOFTWARE, –creating_software=CREATING_SOFTWARE Name of the creating software. Defaults to: *SARMES1* Example: `-s "SARMES1"` Note: the software will be created if it does not exists. Version of the software will be set to a blank string.

-l LICENSE, –license=LICENSE Name of the license. Defaults to: *SAC Commercial License* Example: `-l "SAC Free License"` Note: will be created if it does not exists. License type will be automatically set to *LICENSE_TYPE_ANY* (4)

16.2 Implementation details

Details on the implementation, mainly regarding the source of the data used to populate the catalogue database.

16.2.1 Data and metadata extraction

The ingestion process uses the `dims_lib` package to extract informations from the packages, the following data are extracted and made available for the catalogue:

1. original metadata, is the package's ISOMetadata.xml file
2. thumbnail, read from SacPackage Product folder
3. image, the original SacPackage Product tif image
4. spatial_coverage, this is read from the metadata or directly from the image if it is not found

The following information is read from the ISOMetadata:

product_date '//{xmlns}dateStamp/{xmlns_gco}Date', # Product date

file_identifier '//{xmlns}fileIdentifier/{xmlns_gco}CharacterString',

processing_level_code '{*xmlns*}*processingLevelCode*{xmlns}code/{xmlns_gco}CharacterString',

cloud_cover_percentage '//{xmlns}cloudCoverPercentage/{xmlns_gco}Real',

image_quality_code '{*xmlns*}*imageQualityCode*{xmlns}code/{xmlns_gco}CharacterString',

spatial_coverage '{*xmlns*}*EX_BoundingPolygon*{xmlns_gml}coordinates',

institution_name '//{xmlns}CI_ResponsibleParty/{xmlns}organisationName/{xmlns_gco}CharacterStrin

institution_address '//{xmlns}CI_Address/{xmlns}deliveryPoint/{xmlns_gco}CharacterString',

institution_city '//{xmlns}CI_Address/{xmlns}city/{xmlns_gco}CharacterString',

institution_region '//{xmlns}CI_Address/{xmlns}administrativeArea/{xmlns_gco}CharacterString',

institution_postcode '//{xmlns}CI_Address/{xmlns}postalCode/{xmlns_gco}CharacterString',

institution_country '//{xmlns}CI_Address/{xmlns}country/{xmlns_gco}CharacterString',

Note: The spatial_coverage is first read from *EX_BoundingPolygon* parameter in ISOMetadata.xml, and then read from the geotiff image if not found.

The following information is read from the main image using **GDAL** library:

1. radiometric_resolution
2. band_count
3. spatial_coverage

The following information are reverse-engineered from the **file_identifier**, the process is handled from the function call `GenericSensorProduct.productIdReverse()`

1. acquisition_mode
2. projection
3. path
4. row
5. path shift
6. row shift

Note: The acquisition_mode is a foreign key to the sensors dictionaries, the ingestion process will take care of creating all the necessary entries when they are missing. The mission_group value is not present in **file_identifier** and is hence set to the first type existing in the catalogue, should a new Mission record need to be created during the ingestion process. In practice the Mission dictionary entries should pre-exist and this situation should not arise.

Note: If a new license is created during the ingestion process, the type is defaulted to License.LICENSE_TYPE_ANY

Note: The projection is created if it does not exists, the epsg code is set to 0 by default.