SANSA Catalogue Documentation

Tim Sutton, 2008

Note: This system currently has no automated backup procedure. The main system (Lion) includes around 26TB of online storage that we have no effective way of backing up. This should be addressed as a priority!

Document last update	Document last change
20110412	20110409

Contents

1 Introduction

1.1 Introduction

Note: This section is intended to provide context for the impetus for the development of the catalogue system to third party developers.

1.1.1 The CSIR

This project has been carried out for the erstwhile 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. The CSIR is a large organisation with many divisions of which SAC (Satellite Applications Center) was one. SAC has since been incoporated into the South African National Space Agency (as described below).

1.1.2 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.

1.1.3 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 became part of SANSA as of 1 April 2011 and will and is now SANSA-EO.

1.1.4 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.

1.1.5 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.

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.

The purpose of this document is to provide technical detail covering the setup and deployment of the catalogue, as well as an architectural overview. API documentation

is provided as a separate, complementary document to this one.

2 Installation Guide

2.1 Configuring the database server

The database server is an IBM P5 64bit PPC architecture server. This section details the process for setup and configuration of Debian 4.0 'Woody' onto the server. The machine has since been upgraded to 'Lenny' (Debian 5.0) and will shortly be upgraded to 'Squeeze' (Debian 6.0). Not many linux distributions support PPC64 architecture. Debian is a good choice for this environment because:

- debian itself is an extremely stable and well regarded Linux distribution (it forms the basis of many other distros such as Ubuntu).
- debian supports a wide range of architectures.
- free support is readily available via the debian-ppc channel on IRC (freenode).
- unlike Redhat and Suse, its almost trivial to upgrade from major release to major release and is easy to keep current with security updates.
- there is a huge selection of software packages available easily via the efficient APT package archive so its rarely needed to compile software from source (which introduces software maintenance hassles).

2.1.1 Initial Install

Software Raid Partitioning scheme on p5

1. Server has 4 * 170gb drives.
2. Server has 22gig ram

Reboot server from HMC (Hardware management console)
Press 5 when prompted with a menu so that it will boot from cdrom
Type

```
install64
```

at the boot prompt.

1. Choose install language: English
2. Choose country: South Africa
3. Choose a locale: en_ZA.utf8
4. Choose additional locales: none
5. Keymap for a USB keyboard: American English
6. Primary Network Interface: eth5
7. Network: press cancel durint autoconfigure with dhcp
8. Configure network manually
9. IP Address: 196.35.94.197
10. Netmask: 255.155.155.0
11. Gateway: 196.35.94.1
12. Nameserver: 168.210.2.2
13. Hostname: elephant
14. Domain Name: csir.co.za
15. Partitioning: Manual

We will set up following scheme (identical on each drive)

1. **Primary Partition 1**: 8.2mb bootable flag on, physical volume for PReP ppc boot partition (we will clone this with dd across all devices since it cant reside inside of a software raid device. The PReP partition must reside in the first partition.
2. **Primary Partition 2**: 11.5gb swap (we will stripe swap across all drives for best performance)
3. **Primary Partition 3**: 11 bootable flag on, for / partition - we will use ext3 and clone from sda3 to sdb3,sdc3 and sdd3 after setting up the OS
4. **Primary Partition 4**: 124.8gb All remaining space physical volume for raid (will become part of md0) for /opt partition - we will use raid 5 (stripe with parity) with 3 active drives and 1 hotswap drive.

After doing the above, choose software raid in the partitioning tool and do:

1. **md0**: /, rootfs formated to ext3 Raid 5 with Active drives: sda4, sdb4, sdc4 Hot Spare: sdd4

Save partition layout and continue:

1. Root Password: _____
2. Full Name of new user: Tim Sutton
3. Username for your account: timlinux
4. Password: _____
5. Installing the base system:
6. Use a network mirror? yes (use za mirror as prompted)
7. Choose software to install: Standard system only
8. Device for bootloader installation: /dev/sda1 (we will clone to other drives later with dd)
9. Reboot...

2.1.2 Install openssh

Once the server comes up from its post install reboot you should install key pieces of software on it, starting with ssh (secure shell).

```
sudo apt-get install openssh-server
```

Configure openssh to run on a non-standard port, only accept named users and require public key authentication for added security. To do this edit

```
/etc/ssh/sshd_config
```

And change / add the following values:

```
Port 8697
PermitRootLogin no
#Add additional users who should have ssh access here delimited by spaces
AllowUsers timlinux wluck rgremels cstephens
Protocol 2
ListenAddress 196.35.94.197
RSAAuthentication yes
PubkeyAuthentication yes
PasswordAuthentication no
Banner /etc/sshbanner.txt
```

This forces ssh connections to only allow access to specific users on a non-standard port and using a public/private key pair.

Create this file as /etc/sshbanner.txt

```
              __     __
             /  \~~~/  \
       ,----(     ..    )
      /      \__     __/
     /|         (\  |(
    ^ \   /___\  /\ |   hjw
      |__|   |__|-"    '97


          SAC Database Server

   Access to this computer is restricted to
          authorised personnel.
```

Now restart ssh:

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

2.1.3 Setup system locales

```
sudo dpkg-reconfigure locales
```

- Select 'All Locales'
- Choose en_ZA.utf8 as the default locale
- Choose OK and wait while the locales are generated

2.1.4 Upgrade to lenny

The initial installation above was carried out using Debian 4.0 Stable 'Etch'. Subsequently, Debian 5.0 'Lenny' was released and so I upgraded using the following procedure:

Edit /etc/apt/sources.list as root and replace all 'etch' instances with 'lenny so that it looks like this:

```
#
# deb cdrom:[Debian GNU/Linux 4.0 r4a _lenny_ - Official powerpc NETINST Binary-1 20080804-15:15]/ lenny contrib main

#deb cdrom:[Debian GNU/Linux 4.0 r4a _lenny_ - Official powerpc NETINST Binary-1 20080804-15:15]/ lenny contrib main

deb http://debian.mirror.ac.za/debian/ lenny main
deb-src http://debian.mirror.ac.za/debian/ lenny main

deb http://security.debian.org/ lenny/updates main contrib
deb-src http://security.debian.org/ lenny/updates main contrib
```

Note as well that we have commented out the CD rom sources so that only online apt repositories are used.

Next I ran the following commands to perform the operating system upgrade:

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get dist-upgrade
```

Note this can take some time depending on bandwidth availability.

After the upgrade is completed you should reboot the system so that you are using the newly installed kernel.

2.1.5 Setup Postgres/Postgis

This system is running software RAID 0 (mirror) and is well suited as a database server running postgres and PostGIS (the Spatial data extension for postgres). We will balance the load between the two PPC64 servers so that one is a dedicated database server and the other a dedicated web server. For web pages that are dynamically created, the web server will make database requests off the database server.

```
sudo apt-get install postgresql-8.1 postgresql-8.1-postgis
sudo su - postgres
```

Now as the postgres user make yourself a super user account and a read only user for mapserver:

```
createuser -s -d -r -l -P -E -e timlinux
createuser -S -D -R -l -P -E -e readonly
exit
```

Enter prompts following above commands as needed. Now you have postgres installed and a user created. Next create an empty spatial database:

Note: or see further below to restore existing backups of a db

```
createdb gis
createlang plpgsql gis
```

Now load the postgis sql dump and the srs tables:

```
psql sac < /usr/share/postgresql-8.1-postgis/lwpostgis.sql
psql sac < /usr/share/postgresql-8.1-postgis/spatial_ref_sys.sql
```

From here you can use the shp2pgsql command line tool to load data into postgis, or a tool such as QGIS / udig etc to do it using a gui.

Lastly we need to enable postgis for TCP/IP access. First

```
sudo vim /etc/postgresql/8.1/main/pg_hba.conf
```

And add one entry at the bottom of the file per host that needs access. You can also add a subnet etc. See pg docs for more info on that. I will just add my desktop pc as a client.

```
# Next line added by Tim to enable his desktop machines to connect on TCP/IP
host    all         all         196.35.94.7/32        md5
# Next line added by Tim to enable django server machines to connect on TCP/IP
host    all         all         196.35.94.196/32       md5
```

Also we need to allow tcp/ip connections from hosts other than localhost:

```
sudo vim /etc/postgresql/8.1/main/postgresql.conf
```

And add this line:

```
# Next line added by Tim to enable remote machines to connect on TCP/IP
listen_addresses='*'
```

Next shutdown postgres:

```
sudo /etc/init.d/postgresql-8.1 stop
```

Before we restart, we are going to move the postgres cluster (the files on the filesystem that are used to store the database information) into /opt/ since that opt is mapped to a larger partition, and for backup purposes its nice to have data separate from the main OS partition.

```
sudo mv /var/lib/postgresql /opt/
cd /var/lib/
sudo ln -s /opt/postgresql .
```

Now restart postgres:

```
sudo /etc/init.d/postgresql-8.3 start
```

There are three databases that need to be loaded and regularly backed up and restored:

- gis - All the backdrop gis data e.g. cadastrals, placenames etc
- sac - The web catalogue database used by the django web catalogue
- acs - A clone of the legacy ACS catalogue, ported to Postgresql

To load these databases you should create them and then restore them from a recent backup e.g.:

```
createdb gis
pg_restore gis_postgis_15June2009.dmp |psql gis
createdb sac
pg_restore sac_postgis_15June2009.dmp |psql sac
createdb acs
pg_restore acs_postgis_15June2009.dmp |psql acs
```

For testing, a sac-test database is used which is a snapshot of the sac database:

```
createdb sac-test
pg_restore  sac_postgis_15June2009.dmp |psql sac-test
```

After your databases are loaded you should have a db listing like this:

```
[elephant:timlinux:~] psql -l
List of databases
   Name    |  Owner   | Encoding
-----------+----------+----------
 gis       | timlinux | UTF8
 postgres  | postgres | UTF8
 sac       | timlinux | UTF8
 sac-test  | timlinux | UTF8
 acs       | timlinux | UTF8
 template0 | postgres | UTF8
 template1 | postgres | UTF8
 (7 rows)
```

Finally you need to add a rule in the firewall to allow incoming traffic on port 5432.

Now go on to test with QGIS to see if you can connect ok.

2.1.6 Creating readonly user for mapserver access

You must do this to allow the mapserver client to connect to the database securely (ie. ensuring that no sql injection attacks can take place):

```
grant SELECT on vw_usercart to readonly;
grant select on visit to readonly;
grant select on search to readonly;
```

In addition the gis database needs to have permissions set to read only for all tables for mapserver. Here is a little script I wrote to do that in one go:

```
for TABLE in `echo "\dt" | \
  psql -h 196.35.94.197 -U timlinux gis | \
  awk '{print $3}'`
do
  echo "grant select on $TABLE to readonly;" >> /tmp/script.sql
done
psql -h 196.35.94.197 -U timlinux gis < /tmp/script.sql
```

2.1.7 Nightly clone of the sac catalogue to test

For testing we maintain a clone of the sac catalogue. The clone is replaced nightly. The script to replace the clone is in svn and should be checked out:

```
cd /home/timlinux
svn co https://196.35.94.196/svn/trunk/bash
```

Now edit the crontab:

```
crontab -e
```

And add the clone script to run nightly

```
# 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

#A script to clone the sac live db nightly into the sac test db
#Runs at 2:05 am each night
5 2 * * * /home/timlinux/bash/pg_sac_test_cloner
```

Source of /home/timlinux/bash/pg_sac_test_cloner (replace XXXXXX with a valid password).

```
#/bin/bash

cd /tmp
export PGPASSWORD=XXXXXXX
BACKUP=sac_postgis_`date +%d%B%Y`.dmp
pg_dump -i -Fc -f $BACKUP -x -O sac
dropdb sac-test
createdb sac-test
pg_restore $BACKUP | psql sac-test
rm $BACKUP
echo "SAC Test database successfully cloned from $BACKUP" \
   | mail -s "SAC Test database successfully cloned from $BACKUP" \
   tim@linfiniti.com
```

2.2 Configuring the Lion Catalogue server

The 'Lion' server provides web mapping services for the SAC Catalogue.

It contains the following key components:

- apache web server
- mapserver cgi with ecw support via custom built gdal
- tilecache
- an instance of the django catalogue software used
- IBM informix client sdk software
- the python informix DB adapter

In addition the lion server is connected to two 13TB (effective) Fujitsu Siemens storage arrays.

The Server is an IA 64 processor machine installed with ubuntu server 10.04 LTS. This document assumes a basic Ubuntu Server LTS install has already been carried out.

2.2.1 Package installations

```
sudo apt-get install bc xterm build-essential lxde rpm mc sun-java6-jre \
firefox vim ntpdate lxde xinit rpl django python-django subversion \
utidy python-utidylib python-psycopg2 python-geoip python-django-registration
```

2.2.2 Setup Fibrecat SX60 Storage arrays

Hardware preparation

As root / sudo edit the /etc/fstab and disable the storage array at boot. To do this change this line:

```
UUID=3bdda6ae-3195-47b0-955e-278b5ed51da5 /data    ext3       auto,nouser,noexec,nosuid,rw        1 2
```

to look like this:

```
#UUID=3bdda6ae-3195-47b0-955e-278b5ed51da5 /data    ext3       auto,nouser,noexec,nosuid,rw        1 2
```

The host system was powered off:

```
sudo /sbin/halt
```

The fibrecat system was powered off (note that it has redundant power supplies and both must be powered off).

All drives were then removed and their relative positions and serial numbers were recorded.

The drives were then removed from their caddies and replacement 1.5TB drives were inserted in their place.

After all the replacement drives were inserted, the storage array was powered up again and the system restarted.

We checked that all drives came up properly and no warning lights were displayed.

Configure the storage array in the web admin interface

The array is configured using a web control panel at

```
http://196.35.94.141
```

To login use _____ for username and _____ for password.

Next we had to create a virtual disk - removing the old drives destroys any pre-existing vdisks.

- Click on the **manage** link on the left and then choose **create a vdisk**
- Next choose **Manual Virtual Disk Creation**
- Enter 'sarmesstorage' in the **Enter Virtual Disk Name** box
- For **Select Virtual Disk RAID Level** choose 'Raid 5 - Parity RAID, Parity Distributed'
- Click next to proceed onto disk creation
- Tick all the drives in the enclosure diagram except for the last
- Calculate Formatted Virtual Disk Size of Selected Drives - click this button and verify output is something like that shown below here "'For a RAID Level 5, your selected drives will approximately yield a 15.00 TByte final virtual disk capacity.'"
- For the tickbox **Would you like to add dedicated spare drives for this virtual disk?**, choose Yes
- Click continue
- All the drive bays save the last should now be shown in blue.
- Tick the remaining green bay and then click 'continue' next to the **Add Selected Dedicated Spare Drives to "sarmesstorage" and Continue Creating Virtual Disk:** prompt.

At this stage you will be shown a report that should look something like this:

```
Virtual Disk Name:  sarmesstorage
RAID Level:     5
Virtual Disk Size:  201644.88 GBytes
Drives Chosen:
Serial Number   WWN     Size (GBytes)   Encl.Slot
9VS1J496    5000C50011343DE3    1500.30     0.0
9VS1E8SL    5000C50011229530    1500.30     0.1
9VS1BWB5    5000C5001115F067    1500.30     0.2
9VS1BM5F    5000C5001111301C    1500.30     0.3
9VS1E0WR    5000C500111FC913    1500.30     0.4
9VS1HZ5M    5000C500113537B4    1500.30     0.5
9VS1FWXT    5000C500112BCEA2    1500.30     0.6
9VS1GH9V    5000C500113036EE    1500.30     0.7
9VS1GZMH    5000C5001130D35D    1500.30     0.8
9VS1H6A0    5000C50011353891    1500.30     0.9
9VS1FY6B    5000C500112C210C    1500.30     0.10
Dedicated Spare Drives Chosen:
Serial Number   WWN     Size (GBytes)
9VS1H76L    5000C50011353C6C    1500.30
Virtual Disk Initialization:    Online
```

Now we can proceed to set up partitions ('Volumes') on the virtual disk.

```
Configure Volumes for Virtual Disk sarmesstorage
How Many Volumes    : 1
Create Volumes of Equal Size?
Yes
Expose Volumes to All Hosts?
No
Automatically Assign LUNs?
Disabled
```

```
Would You Like to Name Your Volumes?
No
Advanced Virtual Disk Creation Options  Advanced Options - not used
```

Click 'Create virtual disk' A progress page will appear. Note that the process will take a looooooong time!

Note: It took 3 or 4 days to build the virtual device with 1.5TB disks.

After the virtual disk is built, you need to create a volume mapping.

The volume mapping associates a fibre channel LUN connector to the volume.

In the managment web UI, click: Manage -> volume mapping -> map hosts to volumre.

For the sarmes machine we used the following configuration:

```
Current Host-Volume Relationships


WWN                Host Name       LUN     Port 0 Access   Port 1 Access
10000000C96DABE6   Sarmes1_Port0    0       rw               rw
10000000C961BB34   Sarmes1_Port1    1       rw               rw
All Other Hosts                     None    none             none
```

Note You probably only need to map one WWN / Host / Lun - we think you only need to map Sarmes1_port1 to 10000000C961BB34 but you will need to test experimentally to be sure.

After making these config changes, reboot the sarmes server.

Watch the boot messages or check

```
dmesg
```

You should see a new device listed like this:

```
sd 1:0:0:0: [sdc] Very big device. Trying to use READ CAPACITY(16).
sd 1:0:0:0: [sdc] 29302441984 512-byte hardware sectors (15002850 MB)
sd 1:0:0:0: [sdc] Write Protect is off
sd 1:0:0:0: [sdc] Mode Sense: 93 00 00 08
sd 1:0:0:0: [sdc] Write cache: enabled, read cache: enabled, doesn't support DPO or FUA
sd 1:0:0:0: [sdc] Very big device. Trying to use READ CAPACITY(16).
sd 1:0:0:0: [sdc] 29302441984 512-byte hardware sectors (15002850 MB)
sd 1:0:0:0: [sdc] Write Protect is off
sd 1:0:0:0: [sdc] Mode Sense: 93 00 00 08
sd 1:0:0:0: [sdc] Write cache: enabled, read cache: enabled, doesn't support DPO or FUA
sdc: unknown partition table
sd 1:0:0:0: [sdc] Attached SCSI disk
```

You can see the drive came up as sdc. It pushes the previous sdc drive down to sdd. This is not a problem though since the /etc/fstab uses UUIDs to reference partitions.

Next you can verify this using fdisk:

```
sudo /sbin/fdisk -l /dev/sdc

Disk /dev/sdc: 15002.8 GB, 15002850295808 bytes
255 heads, 63 sectors/track, 1823992 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Disk identifier: 0x00000000

Disk /dev/sdc doesn't contain a valid partition table
```

By default a DOS partition table is used on new devices and by fdisk. One major limitation of this is that it does not support partition sizes greater than 2TB, meaning that most of your large disk device will be inaccessible!

There are two ways two resolve this - using a raw xfs partition (as we have done on SARMES), or using the GPT partition table scheme as we have done on LION).

Creating a large filesystem using GPT

For newer systems we want to use ext4 on a large (non raw) filesystem The kernel must have been compiled with GPT support (it is by default under UBUNTU Jaunty Server Edition >= 9.04). In addition, we need to use **parted** (the command line version of gparted) to format the disk and create the GPT partition table.

In Linux parlance, determining the partition table type is called 'setting the disk label'. In the console transcripts that follow we will set the disk label to GPT, create a large single partition and then format and mount the drive. Once this has been completed, we will use a similar procedure as described above to add an fstab entry so that the volume is mounted at boot time.

This is the procedure I used to create a large ext4 partition using parted:

```
(parted) unit s
(parted) print
Model: FSC FibreCAT_SX1 (scsi)
Disk /dev/sdc: 29302441984s
Sector size (logical/physical): 512B/512B
Partition Table: gpt

Number  Start  End  Size  File system  Name  Flags

(parted) mkpart
Partition name?  []? cataloguestorage2
File system type?  [ext2]? ext4
Start? 34
End? 29302441950
Warning: The resulting partition is not properly aligned for best performance.
Ignore/Cancel? cancel
(parted) mkpart cataloguestorage2 ext4 1 -1
Warning: You requested a partition from 1s to 29302441983s.
The closest location we can manage is 34s to 29302441950s.
Is this still acceptable to you?
Yes/No? yes
Warning: The resulting partition is not properly aligned for best performance.
Ignore/Cancel? Ignore
(parted) p
Model: FSC FibreCAT_SX1 (scsi)
Disk /dev/sdc: 29302441984s
Sector size (logical/physical): 512B/512B
Partition Table: gpt

Number  Start  End           Size          File system  Name              Flags
 1      34s    29302441950s  29302441917s               cataloguestorage2
```

The process sets the drive units to sectors, then creates a new partition leaving 34sectors at the start of the drive.

Now exit parted and create the filesystem:

```
sudo mkfs.ext4 /dev/sdc1
```

Note it will take a little while to process. Finally add a new mount point for the partition and mount it.

```
mkdir /mnt/cataloguestorage2
```

Add an entry to /etc/fstab:

```
/dev/sdc1 /mnt/cataloguestorage2           ext4    relatime,errors=remount-ro       0       2
```

2.2.3 Informix Client Setup

When following the Informix install procedure, do it as root locally on the server since I had problems trying to run the sdk setup tool remotely over an ssh -X connection.

For specific notes on how to set up the client see the informix specific notes (003-3-informix_access.t2t).

2.2.4 Nightly database sync

We should sync the ACS data to our own catalogue database nightly. When all the prerequisites are installed on the Lion server, the updateInformix.sh script can be used to do this on an ad hoc basis. Automating the process requires creation of a cron job:

```
crontab -e
```

Now add the following (adjusing paths if needed):

```
# 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/timlinux/dev/python/sac_catalogue/updateInformix.sh
```

2.2.5 Nightly database backups from elephant

For prudence sake, a nightly dump is made of the databases on ELEPHANT onto the LION server.

```
crontab -e


# Job will run 2:05 am each day
5 2 * * * /home/timlinux/bin/pgbackups
```

The second job described above takes a backup of the gis and catalogue databases on a nightly basis. The pgbackups script looks like this:

```
#/bin/bash

cd /mnt/cataloguestorage/backups/
MONTH=$(date +%B)
YEAR=$(date +%Y)
mkdir -p $YEAR/$MONTH
cd $YEAR/$MONTH
tar cfz opt_`date +%d%B%Y`.tar.gz /opt/ /etc/apache
export PGPASSWORD=pumpkin
pg_dump -i -U timlinux -h elephant -Fc -f gis_postgis_`date +%d%B%Y`.dmp -x -O gis
pg_dump -i -U timlinux -h elephant -Fc -f sac_postgis_`date +%d%B%Y`.dmp -x -O sac
pg_dump -i -U timlinux -h elephant -Fc -f acs_postgis_`date +%d%B%Y`.dmp -x -O acs
psql -h elephant -c "vacuum analyze;" sac
psql -h elephant -c "vacuum analyze;" sac_test
```

To restore you do:

```
createdb sac
createdb gis
pg_restore sac_[filename].dmp | psql sac
pg_restore gis_[filename].dmp | psql gis
pg_restore acs_[filename].dmp | psql acs
```

2.2.6 GDAL and Mapserver Setup

Please see the webmapping chapter (600-webmapping.t2t) for notes on the setup process for GDAL

2.2.7 Apache setup

There are specific notes for the catalogue application in the developer guide. The apache configuration for default (/etc/apache2/sites-available/default) is as listed below.

```
NameVirtualHost *
<VirtualHost *>
  ServerAdmin tim@linfiniti.com
  ServerName 196.35.94.243
  DocumentRoot /var/www/
  <Directory /var/www/>
    Options Indexes FollowSymLinks MultiViews
    AllowOverride None
    Order allow,deny
    allow from all
  </Directory>

  Alias /ss1 /mnt/cataloguestorage/sumbandilasat/SS1
  <Directory /mnt/cataloguestorage/sumbandilasat/SS1>
    Options Indexes FollowSymLinks MultiViews
    AllowOverride None
    Order allow,deny
    allow from all
  </Directory>

  Alias /shade /mnt/cataloguestorage/data/world/aster_dem/final
  <Directory /mnt/cataloguestorage/data/world/aster_dem/final>
    Options Indexes FollowSymLinks MultiViews
    AllowOverride None
    Order allow,deny
    allow from all
  </Directory>

  # Options for fastcgi support:
  # FastCgiConfig -appConnTimeout 60 -idle-timeout 60 -init-start-delay 1 -minProcesses 2 -maxClassProcesses 20 -startDelay 5

  ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
  <Directory "/usr/lib/cgi-bin">
    #Next two lines added by Tim for PyWPS
    SetEnv PYWPS_CFG /etc/pywps.cfg
    SetEnv PYWPS_PROCESSES /opt/wps-processes/sac
    PythonPath "['/opt/','/opt/wps-processes/sac'] + sys.path"
    AllowOverride None
    #Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
    #changed from above for pywps
    Options +ExecCGI -MultiViews +FollowSymLinks
    Order allow,deny
    Allow from all
  </Directory>

  #Alias and dir below added for pywps
  Alias /wps_outputs/ "/tmp/wps_outputs"
  <Directory "/tmp/wps_outputs/">
      Options Indexes MultiViews FollowSymLinks
      AllowOverride None
  </Directory>

  <Location "/sarmes2">
    AuthType Basic
    AuthName "sac"
    AuthUserFile /etc/apache2/dims.passwd
    Require valid-user
```

```
  </Location>


  ErrorLog /var/log/apache2/error.log

  # Possible values include: debug, info, notice, warn, error, crit,
  # alert, emerg.
  LogLevel warn

  CustomLog /var/log/apache2/access.log combined
  ServerSignature On

  # For munin server monitoring
  Alias /munin/ "/var/www/munin/"


  # Reverse proxy to the ordering service on dims
  ProxyRequests Off

  <Proxy *>
  Order deny,allow
  Allow from all
  </Proxy>

  ProxyPass /os4eo http://196.35.94.248:8080/hma/ordering
  ProxyPassReverse /os4eo http://196.35.94.248:8080/hma/ordering


</VirtualHost>
```

This creates various share points through the file system. You should evaluate the file and check that each of the share points listed is indeed present and with the appropriate permissions in the file system.

2.2.8 Proxying Ordering Service Requests

The ordering service on jackal is not a publicly accessible server so we proxy access to it via lion.

```
sudo apt-get install libapache2-mod-proxy-html
sudo a2enmod proxy_http proxy_html headers
```

Now add proxy config to 000-default (as listed in the apache section above).

```
  # Reverse proxy to the ordering service on dims
  ProxyRequests Off

  <Proxy *>
  Order deny,allow
  Allow from all
  </Proxy>

  ProxyPass /os4eo http://196.35.94.248:8080/hma/ordering
  ProxyPassReverse /os4eo http://196.35.94.248:8080/hma/ordering
```

2.2.9 Fibrecat storage arrays config

Catalogue Storage 1

IP Address: 192.168.1.142
Rack Position: **Upper** device as you look at the rack
A WWN: 207000c0ff03a2c3 196.35.94.142 Catalogue Storage

```
RAID Controller B       Yes     Failed  System Detected Failure         862821-0743MV00AK       Down
"cataloguestorage" Volume Information
Number  Name                    LUN     Size (Mbytes)
1       cataloguestorage1       0       15002850
```

```
Status  Size (GB)       Manufacturer Model:Revision     Node WWN Serial Number  Chan:LoopID Port0 Port1         Enclosure
Up      1500.30GB       ATA             ST31500341AS    WWN:5000c5001120e443    SN:9VS1CC29     0:11   ----
```

Catalogue Storage 2 (old Sarmes Storage)

IP Address: 192.168.1.141
A WWN: 207000c0ff0a66c8 196.35.94.141 Sarmes Storage
Rack Position: **Lower** device as you look at the rack

Lion configuration as fibrecat client

```
timlinux@lion:~$ dmesg | grep scsi
[    1.505180] scsi0 : ata_piix
[    1.505252] scsi1 : ata_piix
[    1.506339] scsi2 : ata_piix
[    1.506396] scsi3 : ata_piix
[    1.809264] scsi 2:0:0:0: Direct-Access     ATA      ST3750640NS      n/a  PQ: 0 ANSI: 5
[    1.809395] sd 2:0:0:0: Attached scsi generic sg0 type 0
[    1.813674] scsi 3:0:0:0: Direct-Access     ATA      ST3750640NS      n/a  PQ: 0 ANSI: 5
[    1.813810] sd 3:0:0:0: Attached scsi generic sg1 type 0
[    2.271356] scsi4 : qla2xxx
[    2.630121] scsi5 : qla2xxx
[    3.602342] scsi 4:0:0:0: Enclosure         FSC      FibreCAT_SX1     J200 PQ: 0 ANSI: 4
[    3.603234] scsi 4:0:1:0: Enclosure         FSC      FibreCAT_SX1     J200 PQ: 0 ANSI: 4
[    3.610832] scsi 4:0:0:0: Attached scsi generic sg2 type 13
[    3.610957] scsi 4:0:1:0: Attached scsi generic sg3 type 13
[    3.962310] scsi 5:0:0:0: Direct-Access     FSC      FibreCAT_SX1     J110 PQ: 0 ANSI: 3
[    3.963175] scsi 5:0:1:0: Enclosure         FSC      FibreCAT_SX1     J110 PQ: 0 ANSI: 3
[    3.972099] sd 5:0:0:0: Attached scsi generic sg4 type 0
[    3.972210] ses 5:0:1:0: Attached scsi generic sg5 type 13
```

2.2.10 File System Layout on the catalogue server

The following are the key areas of the file system you should be aware of:

Opt

```
/opt
+-- sac_catalogue
|Â Â  +-- python
|Â Â  +-- sac_live
|Â Â  +-- sac_test
+-- webmapping
 Â Â  +-- config
 Â Â  +-- data -> /mnt/cataloguestorage/data/
 Â Â  +-- fonts
 Â Â  +-- mapfiles
 Â Â  +-- scripts
 Â Â  +-- symbols
 Â Â  +-- templates
```

/usr/local

This is the recommended place in which user compile applications should be installed to. Our installations of gdal, mapserver etc have been placed in this part of the filesystem when installed.

/mnt/cataloguestorage

This is the first of two ˜13TB storage arrays connected to the server. In this storage system, all of the thumbnailas, online remote sensing dataset, backups and data that is being processed are stored.

```
/mnt/cataloguestorage
+-- backups
|Ã Â  +-- 2010
|Ã Â  +-- 2011
+-- data
|Ã Â  +-- africa
|Ã Â  +-- thumbs
|Ã Â  +-- world
|Ã Â  +-- za
+-- imagery_master_copy
|Ã Â  +-- C2B
|Ã Â  +-- S-C
|Ã Â  +-- ZA2
+-- imagery_processing
|Ã Â  +-- cbers
|Ã Â  +-- sacc
|Ã Â  +-- sumbandilasat
+-- mapproxy
|Ã Â  +-- etc
|Ã Â  +-- tmp
|Ã Â  +-- var
+-- thumbnail_processing
|Ã Â  +-- georeferenced_segments_out
|Ã Â  +-- georeferenced_thumbs_out
|Ã Â  +-- segments_out
|Ã Â  +-- thumb_blobs
|Ã Â  +-- to_erase
+-- thumbnails_master_copy
|Ã Â  +-- C2B
|Ã Â  +-- cache
|Ã Â  +-- E1
|Ã Â  +-- E2
|Ã Â  +-- L2
|Ã Â  +-- L3
|Ã Â  +-- L4
|Ã Â  +-- L5
|Ã Â  +-- L7
|Ã Â  +-- N11
|Ã Â  +-- N12
|Ã Â  +-- N14
|Ã Â  +-- N15
|Ã Â  +-- N16
|Ã Â  +-- N17
|Ã Â  +-- N9
|Ã Â  +-- S1
|Ã Â  +-- S2
|Ã Â  +-- S4
|Ã Â  +-- S5
|Ã Â  +-- SACC
|Ã Â  +-- S-C
|Ã Â  +-- ZA2
+-- tilecache
    +-- README
    +-- spot5mosaic10m2007
    +-- spot5mosaic10m2007_4326
    +-- spot5mosaic10m2008
    +-- spot5mosaic10m2008_4326
    +-- spot5mosaic10m2009
    +-- spot5mosaic10m2098_4326
    +-- spot5mosaic2m2007
    +-- spot5mosaic2m2008
    +-- spot5mosaic2m2009
    +-- za_vector
```

2.3 Informix SPOT Catalogue Notes

2.3.1 Accessing the server

```
ssh 196.35.94.210 -l informix
```

Interactive database access:

```
dbaccess
```

2.3.2 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
```

2.3.3 Command line processing using echo

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

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

2.3.4 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
```

2.3.5 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
```

2.3.6 Connecting to the database 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==
```

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 Editition (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
```

2.3.7 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
```

2.3.8 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 GeoObject
         s.  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 07:34:45.6
                        39))


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.
```

2.3.9 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
Sent above and online.log to "Sergio Folco" <sergio.folco.GW-EMI@acsys.it> for diagnostics.

I have stored the logs for issues dating from 7 Jan 2009 in svn under informix_errors/

Following response from ACS, I added a cronjob to do a nightly analyse on the database:

```
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
```

2.3.10 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)
```

2.3.11 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
```

2.3.12 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";
```

2.3.13 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.

2.4 Backup systems

A number of backups are in place but it should be noted that there is cohesive distaster recovery plan and system in place. In particular, there are no full system images, there are no offsite backups and the ~26TB of online storage connected to LION are not backed up. This document describes those backup systems that are in place and that should be checked etc.

We strongly recommend a complete disaster recovery solution be implemented.

2.4.1 Nightly database dumps

2.4.2 Nightly Clone of sac live to sac-test

2.4.3 Nightly thumbs rsync

A duplicate tree of the catalogue thumbnails is stored on cheetah / cxfs at:

```
/cxfs/backups/lion/thumbnails_master_copy/
```

This tree is updated nightly via a cronjob running as the root user on cheetah. The script for the cron job is listed below:

```
#/bin/bash

# This should be run in a cron job nightly by root to
# Pull thumbs over from LION to mirror them on cxfs.
# Root needs to have rssh client key for lion configured - see
# Catalogue documentation for details.
#
# You should add an entry like this to root's crontab to run this every night /
# week as you prefer.
# e.g.
# Job will run 23:05 pm each day
#5 23 * * * /usr/local/bin/sac/mirror_catalogue_thumbs
#
#
cd /cxfs/backups/lion/thumbnails_master_copy/
rsync -ave ssh lion:/mnt/cataloguestorage/thumbnails_master_copy/. .
```

And the cronjob entry itself looks like this.

```
# Job will run at 23:05 each days to maintain a mirror of the catalogue thumbs
5 23 * * * /usr/local/bin/sac/mirror_catalogue_thumbs
```

Note: For the above to work, the filedrop configuration process must have been followed (described elsewhere in this document)

Note: This backup is a mirror only, there is no rotating backup implemented. SAC/SANSA are responsible to ensure that the cxfs is backed up including this directory if proper long term offsite backups are to be had.

2.4.4 Nightly products rsync

Currently the online products **are not backed up**. Ultimately there should be no online products store as everything should be in the DIMS product library, and retrieved via the OGC Ordering Services for EO interface to DIMS.

Until such time, we **highly recommend** making a tape facility available so that occasional snapshots can be made.

2.4.5 Nightly Mapping directory rsync

Currently the mapping data in /mnt/cataloguestorage/data is **not backed up**. This data does not change very often. However it is quite voluminous. We **highly recommend** that SAC should provide access to a tape drive so that occasionally snapshots can be made.

2.4.6 System backups for Elephant

2.4.7 System backups for Lion

2.4.8 Backups pulled to cheetah / cxfs

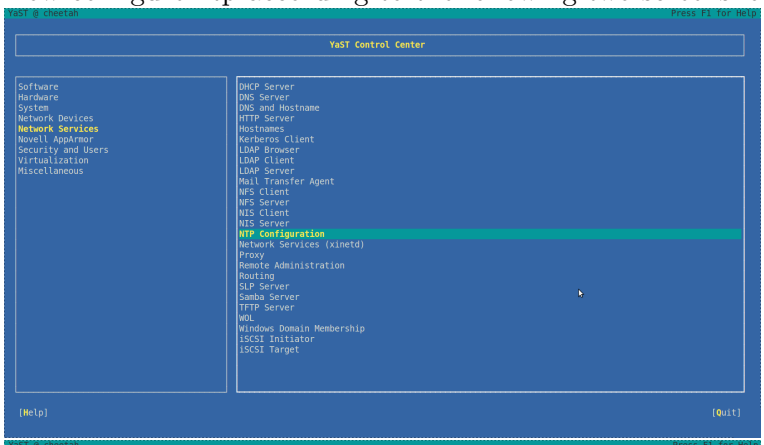
Note: Please see the filedrop chapter which describes how rssh/ssh/filedrop user is set up to allow passwordless scp and rsync operations to be carried out.

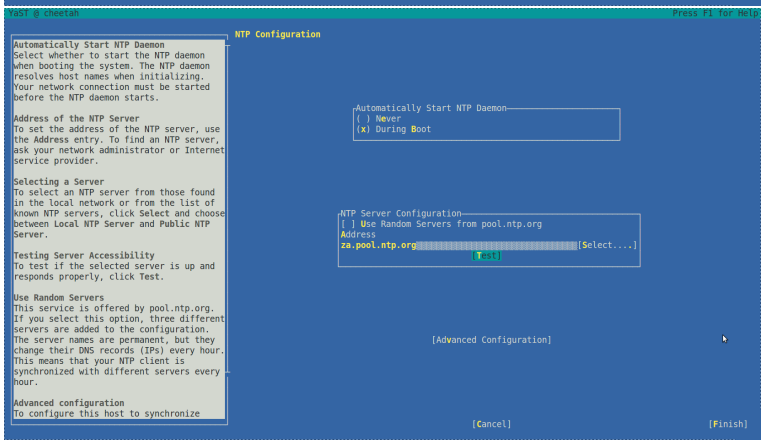
Setup NTP

Firstly, you should ensure that the system time on cheetah is kept properly synchronised via ntp:

```
sudo /sbin/yast2
```

Now configure ntp according to the following two screenshots





Install the backup script

On cheetah the script is available at:

```
#/bin/bash

# This should be run in a cron job nightly by root to
# Pull backups over from LION for the catalogue and GIS databases
# Root needs to have rssh client key for lion configured - see
# Catalogue documentation for details.
#
# You should add an entry like this to root's crontab to run this every night /
# week as you prefer. Note the lion bacup cron job runs at 2:05 am so this
# backup should be timed  for some time later.
# e.g.
# Job will run 7:05 am each day
#5 7 * * * /usr/local/bin/sac/fetch_pg_backups
#
```

```
cd /cxfs/backups/sql_backups/
MONTH=$(date +%B)
YEAR=$(date +%Y)
mkdir -p $YEAR/$MONTH
cd $YEAR/$MONTH
sudo scp lion:/mnt/cataloguestorage/backups/${YEAR}/${MONTH}/sac_postgis_`date +%d%B%Y`.dmp .
sudo scp lion:/mnt/cataloguestorage/backups/${YEAR}/${MONTH}/gis_postgis_`date +%d%B%Y`.dmp .
```

Add crontab entry

As directed abovem you should add a crontab entry e.g.:

```
sudo crontab -e
```

And then add this line:

```
# Job will run 7:05 am each day
5 7 * * * /usr/local/bin/sac/fetch_pg_backups
```

2.5 Redmine Setup

Redmine is an issue tracking tool and is useful in general for project management as well as software development.

Redmine integrates with popular source management systems e.g. git, svn etc.

Note: You should use ruby 1.8! (Unless there is an update on redmine home page allowing otherwise)

```
sudo apt-get install redmine apache2 redmine-sqlite libapache2-mod-fcgid
sudo apt-get install libfcgi libfcgi-dev libfcgi-ruby1.8.1
sudo apt-get install ruby1.8-dev
sudo gem install fcgi
sudo a2enmod fcgid rewrite
```

Add the following to /etc/apache2/mods-enabled/fcgid.config

```
  #Next line added by Tim for Redmine
  SocketPath /tmp/fcgid_sock/


sudo ufw allow 80
sudo ufw status
```

Which should show:

```
Status: active

To                         Action      From
--                         ------      ----
22                         DENY        Anywhere
8697                       ALLOW       Anywhere
80                         ALLOW       Anywhere

25                         ALLOW OUT   Anywhere
```

Set up an apache config like this:

```
# (mod_fastcgi is much harder to configure)
# Configuration for http://localhost:8080
<VirtualHost *>
        ServerAdmin webmaster@localhost
        ServerName redmine.linfiniti.com
        # DefaultInitEnv for module mod_fcgid
        DefaultInitEnv RAILS_RELATIVE_URL_ROOT ""
        DefaultInitEnv X_DEBIAN_SITEID "default"

        # the mod_fcgid socket path
        # Tim: I added this to mods-available/fcgid.conf rather because it was giving an error when included here
        #SocketPath "/var/run/redmine/sockets/default"
        Alias "/plugin_assets/" /var/cache/redmine/default/plugin_assets/
        DocumentRoot /usr/share/redmine/public
        <Directory "/usr/share/redmine/public">
                Options +FollowSymLinks +ExecCGI
                Order allow,deny
                Allow from all
                RewriteEngine On
                RewriteRule ^$ index.html [QSA]
                RewriteRule ^([^.]+)$ $1.html [QSA]
                RewriteCond %{REQUEST_FILENAME} !-f [OR]
                RewriteCond %{REQUEST_FILENAME} dispatch.fcgi$
                RewriteRule ^(.*)$ dispatch.fcgi [QSA,L]
        </Directory>
        ErrorLog ${APACHE_LOG_DIR}/redmine.error.log

        # Possible values include: debug, info, notice, warn, error, crit,
        # alert, emerg.
        LogLevel warn

        CustomLog ${APACHE_LOG_DIR}/redmine.access.log combined
</VirtualHost>


sudo /etc/init.d/apache2 reload
sudo dpkg-reconfigure redmine
```

I used the sqlite backend and took defaults or obvious choices through the package setup.

Default user: admin Default password: admin

2.5.1 Git repository browsing

Create a new project, and then go to the administration panel for it. Then choose

Repositories -> Type: Git

Then enter the url to the .git directory e.g.

```
/opt/git/repositories/sac_catalogue.git/.git
```

2.5.2 Trac to Redmine Migration

Do this before setting up email below.

```
sudo -u www-data X_DEBIAN_SITEID=default RAILS_ENV=production rake -f /usr/share/redmine/Rakefile redmine:migrate_from_trac --trace
```

Output (use similar responses to promptes):

```
** Invoke redmine:migrate_from_trac (first_time)
** Invoke environment (first_time)
** Execute environment
** Execute redmine:migrate_from_trac

WARNING: a new project will be added to Redmine during this process.
Are you sure you want to continue ? [y/N] y

Trac directory []: /opt/trac/sac
Trac database adapter (sqlite, sqlite3, mysql, postgresql) [sqlite]: sqlite3
Trac database encoding [UTF-8]:
```

```
Target project identifier []: sansa-general

Migrating components.............
Migrating milestones..
Migrating custom fields..
Migrating tickets.....................................................................................................................................................................
Migrating wiki.................

Components:      16/16
Milestones:      2/2
Tickets:         337/337
Ticket files:    37/38
Custom values:   443/443
Wiki edits:      18/18
Wiki files:      2/3
```

Mark the old trac as read only (put a note on the front page first):

```
cd /opt
sudo chmod -R ag-w trac
```

2.5.3 Redmine Email Configuration

```
sudo cp /usr/share/redmine/config/email.yml.example /etc/redmine/default/email.yml
```

Now edit that file so it looks like this:

```
production:
  delivery_method: :sendmail
  smtp_settings:
  address: 127.0.0.1
  port: 25

development:
  delivery_method: :sendmail
  smtp_settings:
  address: 127.0.0.1
  port: 25
```

3 Developer Guide

3.1 Working with GIT

3.1.1 Hosting GIT Repos using gitosis

See also http://blog.agdunn.net/?p=277 this. The idea is that we set up a single user account ('git') which proxies commit into the repo and has not shell access. We then add each ssh key for each user that wants access to the repo, assign the user to one or more groups and they can get busy.

Set up the server side

Note: The gitosis repos are hosted on 'orasac1'.
Assuming only your pub ssh key is in authorised keys:

```
sudo apt-get install gitosis
sudo adduser --system --shell /bin/sh --gecos 'git version control' \
             --group --disabled-password --home /opt/git git
sudo -H -u git gitosis-init < /home/timlinux/.ssh/authorized_keys2
```

Allow ssh access to git user

Edit /etc/ssh/sshd_config and add git to AllowUsers list then restart ssh:

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

Check out gitosis admin on the client

```
mkdir -p ~/dev/gitosis
cd ~/dev/gitosis
git clone git@orasac1:gitosis-admin.git sansa-gitosis-admin
```

Adding a user, group and project

```
cd sansa-gitosis-admin
```

Now copy the users pub key into keydir naming it after the user@name listed in their key e.g.:

```
vim keydir/cstephens@cstephens-nb1.pub
```

In this example we are adding access to Casey (I will put him in the admin group too so that he can add more users later). The contents of his keyfile look like this (based on his ssh public key):

```
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEA3WKItKSQU3mtrXwq3P64XM9ChN+exWq69c/q1lWsi/X8yzfbMM9I3Ms
t+wbFmTS4rWWy2xU7H5ES1CeHo9QS6GECMJNSp5EiGSIKCmY3IEwaYuxg4TJVABXwCmJ3MSWRXOFMq1uooaiG4u5SKWQP
JWMgOh41RnEClxLoS2X6sweMPYFlURNTjLja2hCFDROxJaZ2+95/nvjkw2LxihTgeJOGUCmGvGGRHfwZK5HyehqxqZF
LrUjHjtzO5h3yOWgGhdjja12A7RHqLkrZtH7ftY9K8/4O2nAp2IpufWWi6LiGBQrk85bX4JhC/B3lQkbH1MiVM55Dzs
UJFr/xQuw+FQ== cstephens@cstephens-nb1
```

Note: The .pub extension is REQUIRED. Also line breaks added above to prevent page layout issues should not be included.

Then edit gitosis.conf. Add each admin user into the members list for the gitosis-admin group e.g.:

```
[gitosis]

[group gitosis-admin]
writable = gitosis-admin
members = tim@linfiniti.com cstephens@sansa.org.za
```

For a new project, add a new group and repo clause like this:

```
[group sac_catalogue_committers]
writable = sac_catalogue
members = tim@linfiniti.com cstephens@sansa.org.za

[repo sac_catalogue]
gitweb = no
description = SANSA Catalogue
owner = SANSA
daemon = no
```

Note: The .pub extension is not used in the conf file.

We will add entries for the sansa fork of QGIS too (we will need to add Christoph and Riaan to this group).

```
[group sansa_qgis_committers]
writable = sac_catalogue
members = tim@linfiniti.com cstephens@sansa.org.za

[repo qgis]
gitweb = no
description = SANSA Catalogue
owner = SANSA
daemon = no
```

Note: The writeable list in each group should contain a list of those repos that members of that group can write (commit) to. Typically you will want to give the gitosis admin group write permissions to all repos, so each time you add a new repo, update the group like this:

```
[group gitosis-admin]
writable = gitosis-admin sac_catalogue qgis
```

To add a user to an existing project follow the steps above, but just append their key name to the members list.

Now and the new files, commit and push:

```
git add keydir/*
git add gitosis.conf
git commit -m "Set up project for SANSA Catalogue"
git push
```

Create a new empty gitosis hosted repo

The last step is to actually create the repo we defined above (in this case sac_catalogue) and push it up to the server.

```
cd /home/web
mkdir sac_catalogue
cd sac_catalogue
git init
git add .
git remote add origin git@orasac1:sac_catalogue.git
git commit
```

You need to add something to your repo before trying to push it up to the master, so I just put in a .gitignore to start, commit it then push up.

```
touch .gitignore
git add .gitignore
git commit -m "Added ignore file"
git push origin master
```

You should get a message like:

```
Counting objects: 3, done.
Writing objects: 100% (3/3), 221 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
To git@orasac1:sac_catalogue.git
 * [new branch]      master -> master
```

If you got something like this instead:

```
# git push origin master
error: src refspec master does not match any.
fatal: The remote end hung up unexpectedly
```

Its just a symptom that your repo is empty. Add the .gitignore, commit it locally and then try to push it up to the server again.

Note: It is possible to provide anonymous access to this repo too using git-daemon. See the article mentioned at the start of this section for more info.

Checkout

Finally you can check out your repo e.g. on a different computer:

```
git clone git@orasac1:sac_catalogue.git
```

Hosting a clone of an upstream repo in gitosis

Here we want to host a copy of a git repository from upstream and make it available to internal SANSA developers. We will check out the linfiniti git repo in this case:

(On orasac1)

```
sudo su - git
bash
cd repotositories
vim ~/.ssh/config
```

Add the following to the ssh config for git user (we have configured read only access for this user):

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

Now clone the upstream repo:

```
git@orasac01:~/repositories$ git clone --bare git@linfiniti2:qgis.git
```

On your local machine, you can now clone QGIS, work on it, commit your changes to orasac1. When you would like your changes to be merged into QGIS, you email a pull request to linfiniti, we then pull your changes and commit them to svn. To keep the orasac1 copy 'fresh' (synchronised with upstream) a cron job should be configure to pull changes regularly to it.

Check out an existing repo

You should not need to do the above for the QGIS and SANSA Catalogue projects since they already exist and are populated. Above process is for when you want to create a new, empty repo.

The sac_catalogue sources can be checked out like this:

```
git clone git@orasac1:sac_catalogue.git sac_catalogue
```

Similarly the SANSA QGIS fork can be checked out like this:

```
git clone git@orasac1:qgis.git qgis
```

3.1.2 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/

Getting a list of branches

For local branches do:

```
git branch -v
```

For remote branches do:

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

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
```

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
```

Deleting branches

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

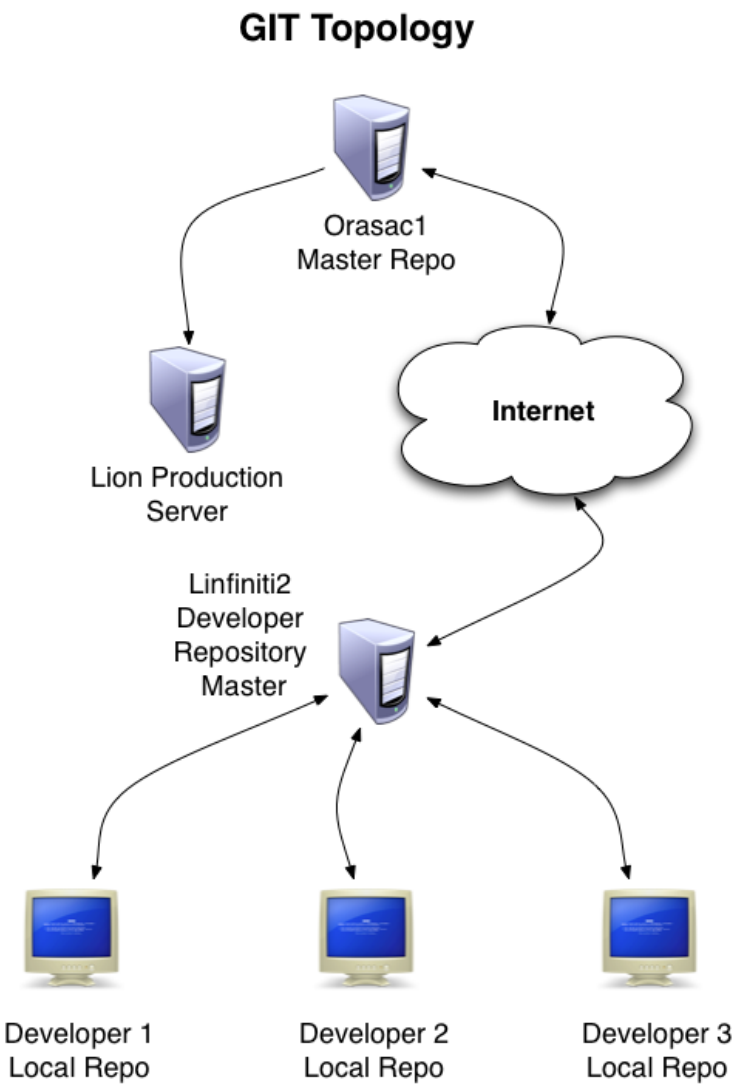
```
git branch -d new-branch
```

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

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

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 git@orasac1:sac_catalogue.git sac_live
git clone git@orasac1:sac_catalogue.git 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.

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
```

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 git@orasac1: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.

3.2 Catalogue Software Installation

3.2.1 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.

Create working dir

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

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
```

3.2.2 Set your ssh config up

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

Host orasac1
  Port 8697
  HostName 196.35.94.196
  FallBackToRsh no
```

3.2.3 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
```

Now you can check out either from linfiniti's repo:

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

or from SANSA's repo:

```
git clone git@orasac1:sac_catalogue.git sac_catalogue
```

Install some development dependencies

```
sudo apt-get install libpq-dev libpq4 libpqxx-dev libxslt1-dev
```

Install easy_install so that we can use pip thereafter:

```
easy_install pip
```

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
```

Note: See

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
```

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
elementsoap
mercurial
#only needed for building api docs
sphinx
```

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.

3.2.4 Source code Check out

Check out this folder using

```
git clone orasac1:sac_catalogue.git 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).

3.2.5 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
```

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.

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
```

3.2.6 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
```

3.2.7 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
```

3.2.8 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
```

3.2.9 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‘
```

3.2.10 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.

3.2.11 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...

3.2.12 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

3.2.13 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"
```

3.2.14 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
```

3.2.15 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
```

3.2.16 set some file permissions

Apache user needs write access in avatars:

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

3.2.17 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
```

3.2.18 Troubleshooting

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.

3.3 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
```

3.3.1 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
```

3.3.2 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
```

3.4 Webmapping

A key part of the system is the provision of web mapping services. These form the backdrops for search and visualisation maps and play various other roles within the system. Although it is not advertised to clients, the web maps can be consumed with any OGC-WMS compatible client via a number of public wms urls.

3.4.1 GDAL Setup

GDAL provides read and write capabilities for numerous GIS formats. We are going to had build GDAL so that we have MrSid and ECW support, as well as support for various other data formats.

Checkout the source

We need to make sure subversion is installed first.

```
sudo apt-get install subversion
```

Make a work directory for building gdal in:

```
cd
mkdir -p dev/cpp
cd dev/cpp
svn co https://svn.osgeo.org/gdal/trunk/gdal gdal-svn
cd gdal-svn
```

3.4.2 Hdf4 and Hdf5 support

```
sudo apt-get install libhdf5-serial-dev libhdf5-serial-1.6.6-0 libhdf4g-dev libhdf4g libhdf4g-doc
```

Building with ECW Support

Note: you can only compress images up to 500mb unless you have an erdas licesne (though as far as I know there is no limit to decompression size).

Install dependencies:

```
sudo apt-get install build-essential libjpeg62-dev libtiff4-dev
```

Go to the erdas ecw sdk web page:
http://www.erdas.com/Products/ERDASProductInformation/tabid/84/CurrentID/1142/Default.aspx
(or just search their site for 'ecw sdk')

Get the ECW JPEG2000 Codec SDK Source Code (second item listed - first if win version)

After the download you should have:

```
ecw_jpeg_2000_sdk_3_3_source.zip
```

Now unzip it:

```
cd /tmp
unzip ecw_jpeg_2000_sdk_3_3_source.zip
unzip ImageCompressionSDKSourceCode3.3Setup_20070509.zip
sudo mv libecwj2-3.3 /usr/local/src/
cd /usr/local/src/
sudo chown -R timlinux libecwj2-3.3/
cd libecwj2-3.3
```

Now build the ecw lib:

```
./configure
make
sudo make install
```

Next gdal must be (re)built with the ecw flag:

Note: You can skip this step if you are going on to add MrSid support below too.

```
cd <gdal src>
make clean
./configure --with-libtiff=internal \
            --with-geotiff=internal \
            --with-ecw=/usr/local \
            --with-python
make -j8
sudo make install
```

See also http://trac.osgeo.org/gdal/wiki/ECW

We configure gdal to build with its own internal copy of libtiff otherwise we encounter problems with assertion errors if the system tiff lib is not compatible. (see http://trac.osgeo.org/gdal/ticket/3139 for more details).

Gdal MrSid Support

You need to first create a user account on the LizardTech website and then get this download (don't worry about the Redhat Enterprise mentioned on the download page, it will work on Ubuntu server too).

http://www.lizardtech.com/developer/members/download.php?dl=Unified_DSDK_8.0_linux.x86-64.gcc41.tgz

Extract the downloaded SDK to /usr/local

To add mr sid support, you need the MrSid geosdk and then to reconfigure and build gdal like this (adjust the Geo_DSDK directory name as needed to match where you extracted it to).

```
cd <gdal src>
make clean
./configure --with-libtiff=internal \
            --with-geotiff=internal \
            --with-ecw=/usr/local \
            --with-python \
            --with-mrsid=/usr/local/Geo_DSDK-7.0.0.2167/ \
            --without-jp2mrs
make -j8
sudo make install
```

3.4.3 Mapserver Setup

First of all you need to compile mapserver from source since we need ecw and MrSid support. We will link it to our hand built GDAL.

```
sudo apt-get build-dep cgi-mapserver libxslt1-dev libpam-dev libreadline-dev
```

Now fetch and install mapserver

```
cd dev/cpp/
wget http://download.osgeo.org/mapserver/mapserver-5.6.5.tar.gz
tar xfz mapserver-5.6.5.tar.gz
cd mapserver-5.6.5/
export LD_LIBRARY_PATH=/usr/local/lib/
```

Now build agg:

```
cd agg-2.4/
./configure
make -j8
sudo make install
```

Now compile mapserver:

```
cd ..
./configure \
    --prefix=/usr \
    --enable-debug \
    --without-tiff \
    --without-pdf \
    --with-gd=/usr \
    --with-freetype=/usr \
    --with-zlib=/usr \
    --with-png=/usr \
    --with-xpm=/usr \
    --with-jpeg=/usr \
    --with-gdal=/usr/local/bin/gdal-config \
    --with-ogr \
    --with-proj \
    --with-eppl \
```

```
    --with-postgis \
    --with-wcs \
    --with-wms \
    --with-wmsclient \
    --with-wfs \
    --with-wfsclient \
    --with-threads \
    --with-geos \
    --with-fastcgi \
    --with-agg=/usr/local
apt-get remove cgi-mapserver
make -j8
sudo cp mapserv /usr/lib/cgi-bin/
```

Once it is built you can check if everything is ok by doing:

```
/usr/lib/cgi-bin/mapserv -v
```

Which should give out something like:

```
MapServer version 5.0.3 OUTPUT=GIF OUTPUT=PNG OUTPUT=JPEG OUTPUT=WBMP
OUTPUT=SVG SUPPORTS=PROJ SUPPORTS=AGG SUPPORTS=FREETYPE SUPPORTS=WMS_SERVER
SUPPORTS=WMS_CLIENT SUPPORTS=WFS_SERVER SUPPORTS=WFS_CLIENT SUPPORTS=WCS_SERVER
SUPPORTS=FASTCGI SUPPORTS=THREADS SUPPORTS=GEOS INPUT=EPPL7 INPUT=POSTGIS
INPUT=OGR INPUT=GDAL INPUT=SHAPEFILE
```

Ensure that after doing the above, your mapserver install supports fastcgi:

```
SUPPORTS=FASTCGI
```

3.4.4 Set up the web mapping dir

The web mapping dir contains mapserver configuration files and resources. It should be checked out of GIT first. You should make sure you have needed permissions for checking the repo out into /opt.

```
cd /opt
git clone git@orasac1:sansa_webmapping.git webmapping
```

After checkout of the webmapping GIT project, you should create a symlink to the data directory:

```
cd webmapping
ln -s /mnt/cataloguestorage2/gisdata/ data
```

3.4.5 Apache configuration

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

Add these map file paths to your /etc/apache2/sites-enabled/default file:

```
Include /opt/webmapping/apache-include/mapserver.conf
```

3.4.6 Mapserver database connection details encrytion

To allow storing mapfiles with database connection details in git, they should be hashed using the [documented msencrypt utility http://mapserver.org/utilities/msencrypt.html].

```
/home/timlinux/mapserver-5.4.2/msencrypt -keygen mapserver-key.txt
/home/timlinux/mapserver-5.4.2/msencrypt -key mapserver-key.txt "fooo"
```

Foo being the username and password for the databse connection. This produces output like this.

‘688477225F5ABDFA‘

Then in the mapfile replace all usernames and passwords with hashes and freely commit the mapfiles to git.

At the top of each mapfile just after the MAP clause add a line like this:

```
CONFIG "MS_ENCRYPTION_KEY" "/opt/webmapping/mapfiles/mapserver-key.txt"
```

And for the connection string to the database use this format:

```
CONNECTION "user={688477225F5ABDFA} password={688477225F5ABDFA} dbname=sac host=localhost"
```

3.4.7 Adding a new backdrop layer

In this section we will use the new 2010 ZA SPOT5 Mosaic as an example of how to deploy a new backdrop layer into the WMS environment.

Synchronising the data to the catalogue server

The first task it to place the data onto the catalogue server (in our case LION). The data could be copied over by various means. In the SAC context it will typically come from the CXFS file store:

on cheetah:

```
cd /cxfs/archive/production/Spot5_Mosaic_Packaging_2010/Imagery/Latlong/TLTIF
dmfind . -state MIG -o -state OFL | dmget
rsync -ave ssh *.gz lion:/mnt/cataloguestorage2/gisdata/za/SPOT5_2010/tif-gz/
dmput *
```

The second command above instructs to retrieve all migrated files from tape, which is needed before the data can be synced over to LION via rsync over ssh.

The last command instructs dmf to push the files back down to tape again when we are done.

Batch conversion to ecw

To covert the mosaic tiles in batch we use a simple script like this (called tif2ecw):

```
#!/bin/bash
mkdir ecw
EXT=tif
for FILE in tif/*.${EXT}
do
  BASENAME=$(basename $FILE .${EXT})
  OUTFILE=ecw/${BASENAME}.ecw
  LOCKFILE=${BASENAME}.lock
  echo "Processing: ${BASENAME}.${EXT}"
  if [ -f $LOCKFILE ] || [ -f $OUTFILE ] #skip if exists
  then
    echo "Skipping: $OUTFILE"
  else
    /usr/local/bin/gdal_translate -of ECW -co LARGE_OK=YES $FILE $OUTFILE
    rm $LOCKFILE
  fi
done
```

Note the above script requires that you have an appropriate ERDAS license so that you can compress large files (although it is not enforced anywhere).

So assuming our data exist in /mnt/cataloguestorage2/gisdata/za/SPOT5_2010/tif-gz/ and we run the script from /mnt/cataloguestorage2/gisdata/za/SPOT5_2010/, the result will be a new directory: /mnt/cataloguestorage2/gisdata/za/SPOT5_2010/ecw containing the mosaic tiles in ecw format.

Creation of a virtual raster

Gdal has a concept called \\virtual rasters\\ (roughly analagous to ESRI image catalogues I believe) that presents multiple images as a single file resource.

We will create a virtual raster for the mosaic so that we can simply add a single layer to our mapfile configuration.

```
cd ecw
gdalbuildvrt za2010ecw.vrt *.ecw
```

Creation of a new mapfile

Each new backdrop layer has a new mapfile created for it. The mapfiles are stored in the GIT repository with passwords and usernames tokenised. When checked out, mapfiles are placed in the webmapping directory under:

```
/opt/webmapping/mapfiles
```

The structure of this directory looks like this:

```
webmapping/
+-- apache-include
|   +-- README
|   \-- mapserver.conf
+-- config
|Ã Â  +-- 000-default
|Ã Â  +-- default
|Ã Â  \-- tilecache.cfg
+-- data -> /mnt/cataloguestorage/data/
+-- fonts
|Ã Â  +-- arialbd.ttf
|Ã Â  +-- arialbi.ttf
|Ã Â  +-- ariali.ttf
|Ã Â  +-- arial.ttf
|Ã Â  +-- ariblk.ttf
|Ã Â  +-- courbd.ttf
|Ã Â  +-- courbi.ttf
```

```
|Ã Â  +-- couri.ttf
|Ã Â  +-- cour.ttf
|Ã Â  +-- fonts.list
|Ã Â  \-- symbols.ttf
+-- mapfiles
|Ã Â  +-- geonames.map
|Ã Â  +-- cart.map
|Ã Â  +-- debug.include
|Ã Â  +-- jpl.include
|Ã Â  +-- landsat7-global-mosaic.map
|Ã Â  +-- pg.map
|Ã Â  +-- searches.map
|Ã Â  +-- vector_layers.map.include
|Ã Â  +-- visitors.map
|Ã Â  +-- world.map
|Ã Â  +-- za.map
|Ã Â  +-- za_nbi.map
|Ã Â  +-- za_vector.map
|Ã Â  +-- za_vector_spot2007_10m_ecw.map
|Ã Â  +-- za_vector_spot2007_2_5m_ecw.map
|Ã Â  +-- za_vector_spot2008_10m_ecw.map
|Ã Â  +-- za_vector_spot2008_2_5m_ecw.map
|Ã Â  +-- za_vector_spot2009_10m_ecw.map
|Ã Â  +-- za_vector_spot2009_2_5m_ecw.map
|Ã Â  +-- za_vector_spot2010_2_5m_tif.map
|Ã Â  \-- za_vector_test.map
+-- scripts
|Ã Â  \-- reseed.sh
+-- symbols
|Ã Â  +-- flaeche1_1.png
|Ã Â  +-- flaeche1.png
|Ã Â  +-- flaeche2_1.png
|Ã Â  +-- flaeche2.png
|Ã Â  +-- flaeche3.png
|Ã Â  +-- schraffur.png
|Ã Â  +-- stern.png
|Ã Â  +-- symbols.sym
|Ã Â  \-- welle.png
\-- templates
    +-- search_footer.html
    +-- search_header.html
    +-- search.html
    \-- search.txt
```

So under /opt/webmapping/mapfiles we simply copy the mapfile definition from a previous year. Of course depending on what you are trying to achieve, you could create a totally new map file in this case too.

```
cp za_vector_spot2009_2_5m_ecw.map za_vector_spot2010_2_5m_ecw.map
```

For clarity it is suggested to stick to a standardised naming convention.

Now the map file needs to be edited and the relevant part for the mosaic layer specified.

```
#
# Notes:
#
# Tim Sutton 2009
#
# By using status default for all layers, mapserver will render them
# all based on their scale dependent ranges when open layers
# makes a request. This is a good thing since It will allow us
# to create complex maps without having to add many layer definitions to
# OpenLayers.
#

MAP
  NAME "SouthAfricaSPOT5Mosaic2010_2.5m"
  # Map image size
  SIZE 400 400
  UNITS dd

  #EXTENT 28.156069 -25.890870 28.169983 -25.879721
  EXTENT -180 -90 180 90
  PROJECTION
    "init=epsg:4326"
  END
```

```
SYMBOLSET "../symbols/symbols.sym"
FONTSET "../fonts/fonts.list"
# Background color for the map canvas -- change as desired
IMAGECOLOR 192 192 192
IMAGEQUALITY 95
# Background color for the map canvas -- change as desired
IMAGECOLOR -1 -1 -1
IMAGEQUALITY 95

INCLUDE "debug.include"

#IMAGETYPE png24
#OUTPUTFORMAT
#  # use the new rendering backend from MapServer 5
#  NAME 'AGGA'
#  DRIVER AGG/PNG
#  IMAGEMODE RGBA
#END

#OUTPUTFORMAT
#  NAME png
#  DRIVER 'GD/PNG'
#  MIMETYPE 'image/png'
#  IMAGEMODE PC256
#  EXTENSION 'png'
#END

IMAGETYPE jpeg
OUTPUTFORMAT
  # use the new rendering backend from MapServer 5
  NAME 'AGG_JPEG'
  DRIVER AGG/JPEG
  IMAGEMODE RGB
END

# Legend
LEGEND
    IMAGECOLOR 255 255 255
  STATUS OFF
  KEYSIZE 18 12
  LABEL
    TYPE BITMAP
    SIZE MEDIUM
    COLOR 0 0 89
  END
END

# Web interface definition. Only the template parameter
# is required to display a map. See MapServer documentation
WEB
  # Set IMAGEPATH to the path where MapServer should
  # write its output.
  IMAGEPATH '/tmp/'

  # Set IMAGEURL to the url that points to IMAGEPATH
  # as defined in your web server configuration
  IMAGEURL '/tmp/'

  # WMS server settings
  METADATA
    'wms_title'           'South Africa SPOT 5 Mosaic 2010 2.5m'
    'wms_onlineresource'  'http://196.35.94.243/cgi-bin/mapserv?map=ZA_SPOT2010&'
    'wms_srs'             'EPSG:4326 EPSG:900913'
  END

  #Scale range at which web interface will operate
  # Template and header/footer settings
  # Only the template parameter is required to display a map. See MapServer documentation
END

LAYER
  NAME 'BlueMarble'
  TYPE RASTER
  DATA '/opt/webmapping/data/world/bluemarble/rectifywesternhemisphere.tif'
  METADATA
    'wms_title' 'WesternHemisphere'
    'wms_srs'             'EPSG:4326'
  END
  STATUS DEFAULT
  TRANSPARENCY 100
  PROJECTION
  'proj=longlat'
  'ellps=WGS84'
```

```
    'datum=WGS84'
    'no_defs'
    ''
    END
  END

  LAYER
    NAME 'BlueMarble'
    TYPE RASTER
    DATA '/opt/webmapping/data/world/bluemarble/rectifyeasternhemisphere.tif'
    METADATA
      'wms_title' 'EasternHemisphere'
      'wms_srs'             'EPSG:4326'
    END
    STATUS DEFAULT
    TRANSPARENCY 100
    PROJECTION
    'proj=longlat'
    'ellps=WGS84'
    'datum=WGS84'
    'no_defs'
    ''
    END
  END

  LAYER
    NAME "Jpl"
    TYPE RASTER
    CONNECTION "http://wms.jpl.nasa.gov/wms.cgi?"
    CONNECTIONTYPE WMS
    METADATA
      "wms_srs"             "EPSG:4326"
      "wms_name"            "global_mosaic_base" #comma separated list of layer names
      "wms_server_version"  "1.1.1"
      "wms_format"          "image/png"
      #"wms_auth_username" "username"
      #"wms_auth_password" "password"
      "wms_bgcolor" "0xFFFFFF"
    END
  END

  LAYER
    NAME 'Spot5_RSA_2009_2_5m'
    TYPE RASTER
    DATA '/opt/webmapping/data/za/SPOT5_2010/ecw/za2010ecw.vrt'
    METADATA
      'wms_title' 'Spot5_RSA_2010_2_5m'
      'wms_srs'             'EPSG:4326'
    END
    STATUS DEFAULT
    TRANSPARENCY 100
    OFFSITE 0 0 0 #transparent pixels
    MAXSCALEDENOM 1000000
    MINSCALEDENOM 0
    PROJECTION
    'proj=longlat'
    'ellps=WGS84'
    'datum=WGS84'
    'no_defs'
    ''
    END
  END

  INCLUDE 'vector_layers.map.include'
END
```

This file should be committed in GIT under 'webmapping/mapfiles' in the GIT repo.

Note: A detailed description of the above file is beyond the scope of this document - please see the UMN Mapserver project for full documentation.

Deploy under apache

To deploy your web mapping project, you need to do two things:

1. Add an entry to mapserver.conf
2. Reload apache

To add a new entry to mapserver.conf, edit apache-include/mapserver.conf and add a new line e.g.

```
SetEnv ZA_SPOT2010 "/opt/webmapping/mapfiles/za_vector_spot2010_2_5m_ecw.map"
```

Then test the configuration and reload:

```
sudo apache2ctl -t
```

If it reports ok then reload:

```
sudo /etc/init.d/apache2 reload
```

Testing

You can test using a WMS client (QGIS!) by adding a new connection to the url e.g.:

```
http://196.35.94.243/cgi-bin/mapserv?map=ZA_SPOT2010
```

3.4.8 Available end points

You can connect to the various end points / map services using the following urls:

```
http://196.35.94.243/cgi-bin/mapserv?map=ZA_VECTOR
http://196.35.94.243/cgi-bin/mapserv?map=GEONAMES
http://196.35.94.243/cgi-bin/mapserv?map=ZA_VECTOR_TEST
http://196.35.94.243/cgi-bin/mapserv?map=ZA_SPOT2007
http://196.35.94.243/cgi-bin/mapserv?map=ZA_SPOT2008
http://196.35.94.243/cgi-bin/mapserv?map=ZA_SPOT2009
http://196.35.94.243/cgi-bin/mapserv?map=ZA_SPOT2010
http://196.35.94.243/cgi-bin/mapserv?map=ZA_SPOT10m2007
http://196.35.94.243/cgi-bin/mapserv?map=ZA_SPOT10m2008
http://196.35.94.243/cgi-bin/mapserv?map=ZA_SPOT10m2009
http://196.35.94.243/cgi-bin/mapserv?map=ZA_SPOT10m2010
http://196.35.94.243/cgi-bin/mapserv?map=L7_MOSAIC
http://196.35.94.243/cgi-bin/mapserv?map=ZA_NBI
http://196.35.94.243/cgi-bin/mapserv?map=WORLD
http://196.35.94.243/cgi-bin/mapserv?map=SEARCHES
http://196.35.94.243/cgi-bin/mapserv?map=CART
http://196.35.94.243/cgi-bin/mapserv?map=VISITORS
```

Please exercise discretion as to which of these you make publicly available - there is currently no access control and not all of these datasets may be made available to the general public.

Please note these end points will change when the server is moved into the DMZ.

3.4.9 Issue Tracking

There is an issue tracker for web mapping related issues here:

```
http://196.35.94.196/projects/sansa-webmapping
```

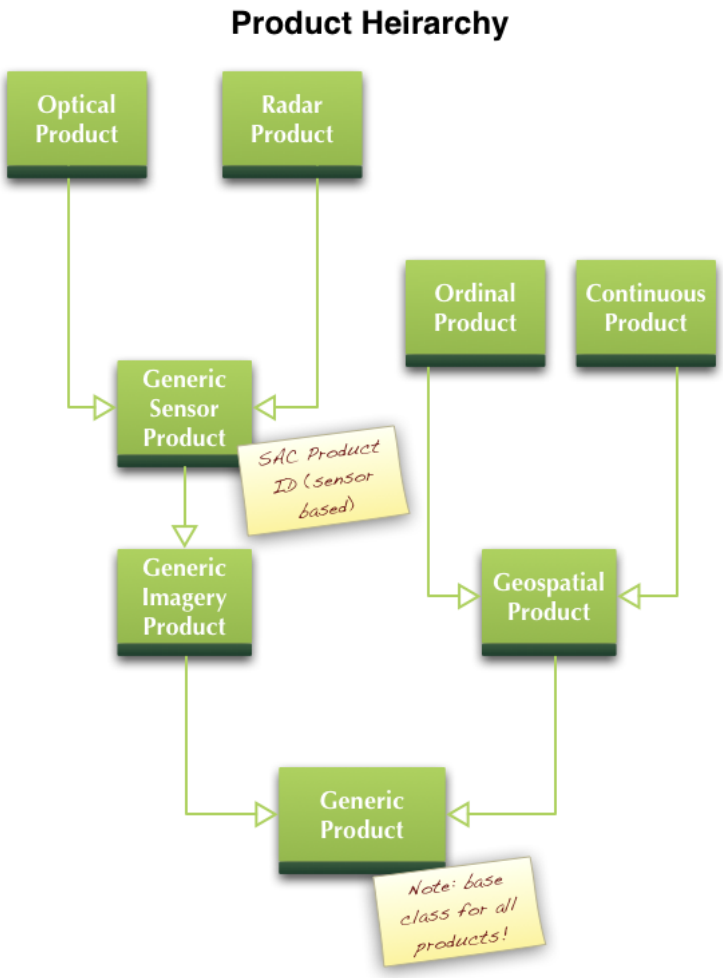
4 Schema

4.1 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.

4.1.1 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.

Generic Product Properties

The following properties are defined for generic products:

Property	Type
product_date	models.DateTimeField
processing_level	models.ForeignKey
owner	models.ForeignKey
license	models.ForeignKey
spatial_coverage	models.PolygonField
projection	models.ForeignKey
quality	models.ForeignKey
creating_software	models.ForeignKey
original_product_id	models.CharField
product_id	models.CharField
product_revision	models.CharField
local_storage_path	models.CharField
metadata	models.TextField
remote_thumbnail_url	models.TextField
concrete	False

Product Date is the generalised date for the product. This may be equivalent to the product_acquisition_start of sensor based products, the mid-point between product_acquisition_start and product_acquisition_end for sensor based products, or an arbitrary date assigned by an operator at point of product ingestion. Product Date is used as the basis for any date-centric search activities in the catalogue.

Processing Level. All products have a descriptor indicating to what level they have been processed. For sensor based products, this level may indicate whether the product has been georeferenced or not. For Generic Spatial Products, it may indicate what kind of analysis was carried out in order to create the product. *see:* dictionaries section below.

Local Storage Path

```
sac=# select local_storage_path from catalogue_genericproduct where local_storage_path is not null limit 5;
local_storage_path
-------------------------------------------------------------------------------
ZA2/1Ab/2010/5/31/ZA2_MSS_R3B_FMC4_080E_57_026N_47_100531_033947_L1Ab_ORBIT-.tif.bz2
ZA2/1Ab/2010/3/31/ZA2_MSS_R3B_FMC4_176E_47_039S_35_100331_204030_L1Ab_ORBIT-.tif.bz2
```

The path is relative to the IMAGERY_ROOT defined in settings.py and is in the form:
mission/processing level/yyyy/m/d/product_id.¡file extension¿.bz2
or
mission/processing level/yyyy/m/d/product_id.¡file extension¿.tar.bz2 (for multifile products)

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.

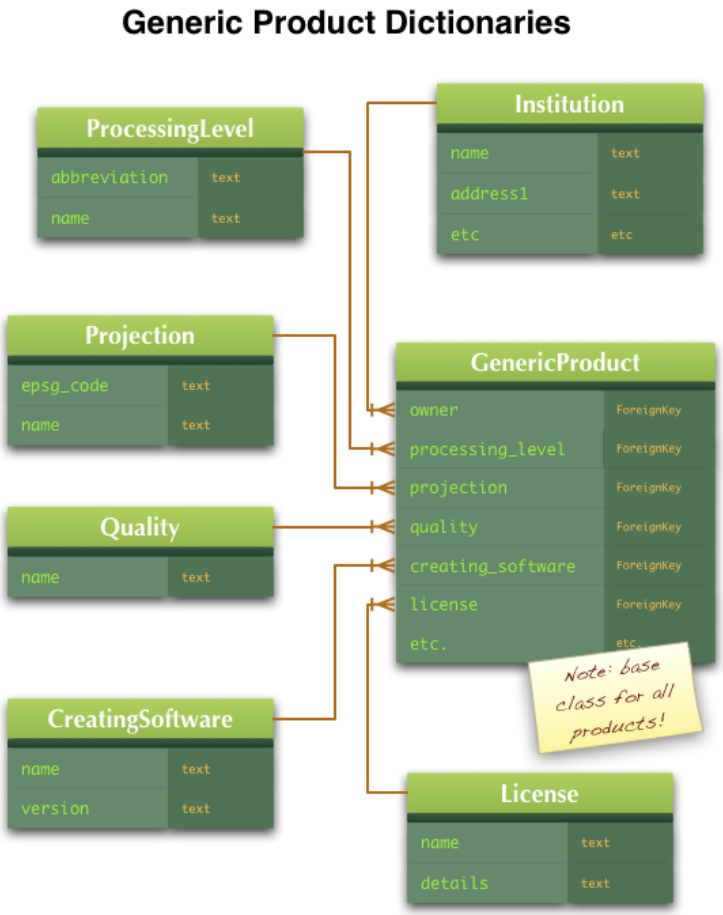
Since this is an abstract class, it has no direct naming scheme of its own.

———— **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.

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

Constraint: Name must be unique.

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

Constraints: Name and abbreviation must both be unique.

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

Constraints: EPSG Code must be unique.

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

Constraints: Name must be unique.

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	

Constraints: Name must be unique.

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?

4.1.2 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).

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. The follow section was from Wolfgang's notes but it does not model well:

- composite products my span multiple modes, types, sensors and even missions.
- scenes may span long time periods and the concept of a central scene is ambiguous
- products may span larger areas than a single QDS map and in other countries mapping units may differ.

- composite products may include other imagery types eg. aerial photos.

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: be `MT_yymmdd_yymmdd_SSS_sss_ttt_mmmm_pppp_ps_rrrr_rs_yymmdd_hhmmss_`
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 (k)
ps	path shift
rrrr	row (j)
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

Imagery product Properties

A GenericImageryProduct extends the generic product model with these properties:

- geometric_resolution
- geometric_resolution_x
- geometric_resolution_y

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).

Dictionaries

No additional dictionaries are introduced with this class.

4.1.3 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.

Product ID Naming Scheme

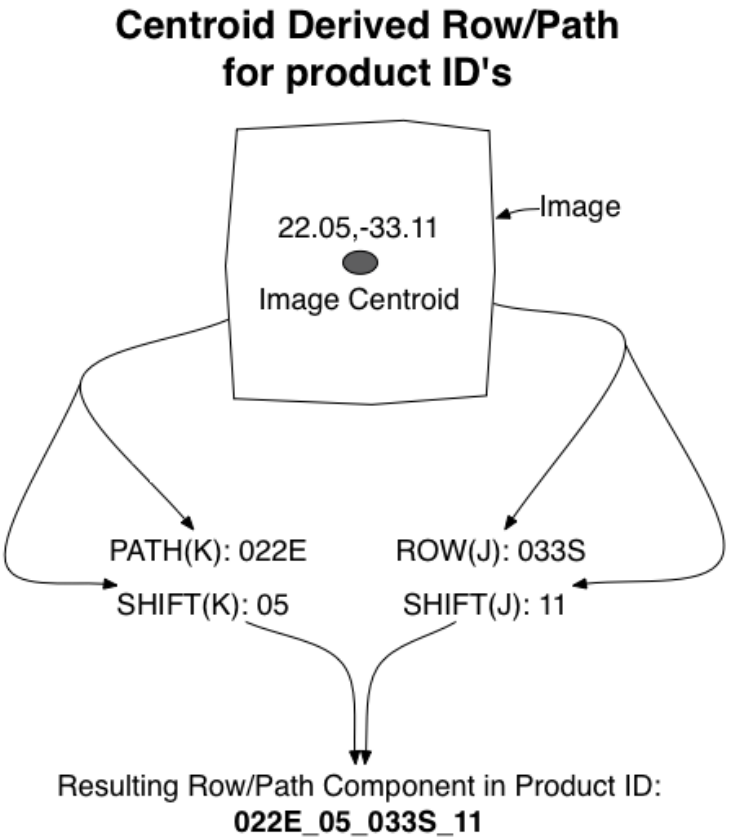
The following scheme is used for assigning product id's for sensor based products:
Single scene file names:
`SSS_sss_ttt_mmmm_pppp_ps_rrrr_rs_yymmdd_hhmmss_LLLL_PPPPPP`
e.g. L7-_ETM_HRF_SAM-_168-_00_077-_010530_–_L3Ab_UTM36S
The following key can be used to decode the above:

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 (k)
ps	path shift
rrrr	row (j)
rs	row span
yymmdd	date
hhmmss	time
LLLL	processing Level code eg. L2A; L4Ab
PPPPPP	Projection eg. UTM35S; LATLON; ORBIT-

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 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.

Where Row and Path are not applicable for a sensor (e.g. Sumbandilasat, some Radar sensors), a centroid derived Path, Path offset, Row and Row offset will be assigned as per the illustration below:

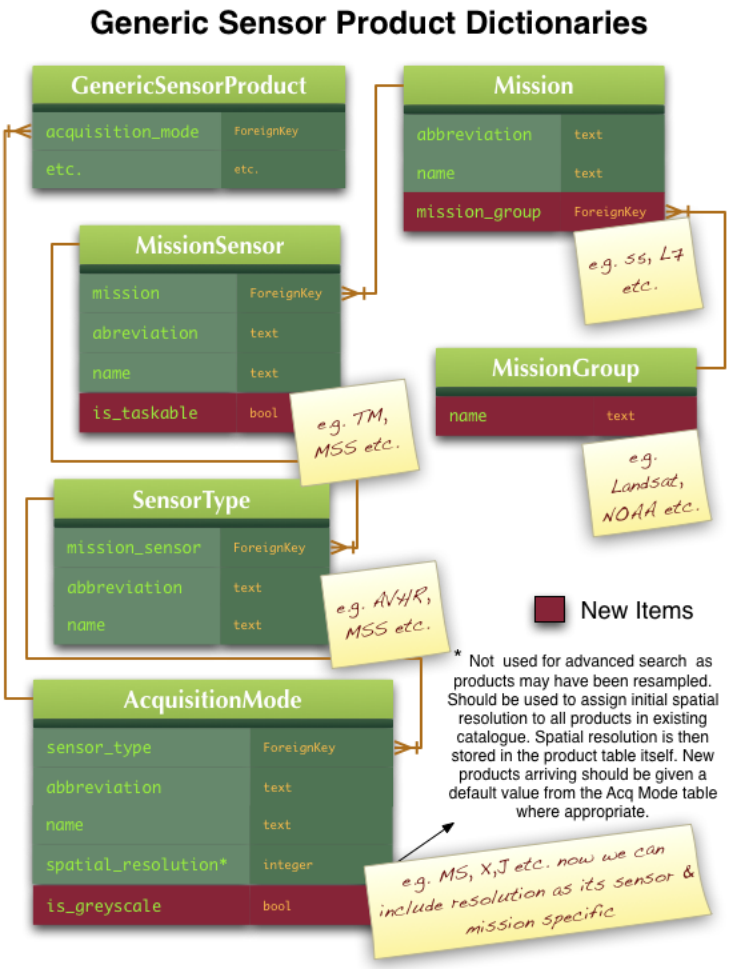


Generic Sensor Product Aggregation Rules

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

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

The mission abbreviation and name must be unique.

Note: Wolfgang we are using ZA2 instead of SS here.

Note: RE - RapidEye needs to be added to this list

Constraints: Name and abbreviation must be unique_together. Name must be unique.

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

The abbreviation field and the foreign key to Mission must be unique together.

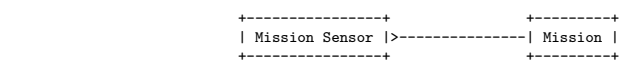
Note: Ale, I would like to get rid of the has data field if poss. It is used to restrict the display of sensors to users to only those with data associated to them. Is there a

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 Multispectral

Constraints: Name and abbreviation must be unique_together. Name must be unique.

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	

The abbreviation field and the foreign key to MissionSensor must be unique together.

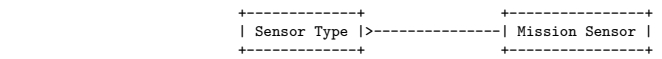
Note: Wolfgang please populate the related mission sensor(s) per id above.

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	

The abbreviation field and the foreign key to SensorType must be unique together.

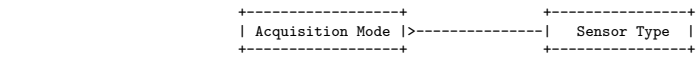
Note: Wolfgang to populate sensor type. Entries should be duplicated for each sensor type that has that mode e.g.

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).

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.

====== Dictionaries naming rules ======

Given the relationships above mentioned for sensors dictionaries, the following rules are applied in cascade mode in order to build a unique dictionary item in the tree:

Dictionary	Naming scheme	Constraints (unique together if in brackets)	Example
Mission	abbreviation	abbreviation, name	L5
MissionSensor	abbreviation:Mission	abbreviation,	TM:L5
SensorType	abbreviation:MissionSensor	abbreviation,	MST:TM:L5
AcquisitionMode	abbreviation:SensorType	abbreviation,	HRT:MST:TM:L5

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.

4.1.4 Optical products

Synopsis: Imagery where each pixel represents the reflectance value of light waves reflected from a remote target within a given segment of the light spectrum.

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.

Optical Product Properties

The optical product model introduces a number of properies in addition to those inherited from the GenericSensorProduct base class.

- cloud_cover
- sensor_inclination_angle
- sensor_viewing_angle
- gain_name
- gain_value_per_channel
- gain_change_per_channel
- bias_per_channel
- solar_zenith_angle
- solar_azimuth_angle
- earth_sun_distance

Product ID Naming Scheme

The naming of optical products follows the rules from its base class, GenericSensorProduct.

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**.

4.1.5 Radar Products

Synopsis: Imagery where each pixel represents the reflectance value of radio waves reflected from a remote target.
Concrete or Abstract: Concrete

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**.

Product ID Naming Scheme

Follows the same scheme as defined in OpticalProduct documentation.

Optical Product Properties

The radar product model introduces a number of properies in addition to those inherited from the GenericSensorProduct base class. These properties are not shared with optical products.

- imaging_mode
- look_direction
- antenna_receive_configuration
- polarising_mode
- polarising_list
- slant_range_resolution
- azimuth_range_resolution
- orbit_direction
- calibration
- incidence_angle

4.1.6 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.

Geospatial Product Properties

All geospatial products share the following properties:

- name
- description
- processing_notes
- equivalent_scale
- data_type
- temporal_extent_start
- temporal_extent_end
- place_type
- place
- primary_topic
- tags

Properties from GenericProduct are also inherited.

The **name** property is a descriptive name (in plain english) for the dataset e.g. "Land-cover map of South Africa, 2010"

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.

The **processing notes** property is a freeform text field where any specific notes and logging information relating to the processing of the dataset can be recorded. This data should be provided in plain text format.

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. A default of 1000000 (one million) shall be assigned to any dataset where the scale is not explicitly defined.

The **data_type** property describes what type of data is represented in the dataset. The options for data_type are defined in a dictionary in the GeospatialProduct model viz:

Abbreviation	Name
RA	Raster
VP	Vector - Points
VL	Vector - Lines
VA	Vector - Areas / Polygons

Note: The ISO-19115 spec defines an MD_SpatialRepresentationTypeCode but this is not granular to the feature type level. As such a programmatic mapping shall be such that types VP/VL/VA are mapped to MD_SpatialRepresentationTypeCode 'vector 001' and RA is mapped to MD_SpatialRepresentationTypeCode 'grid'.

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 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. If not specified, the temporal extent **start** shall default to the current data and time at the moment of product ingestion / creation. If not specified, the temporal extent **end** shall default to be the same value as the temporal extent start.

The **place** and **place_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 single geographic place. These elements must be defined by the script performing ingestion which is out of scope for this work package. As such the only current way to add new geospatial products is manually through the admin panel, or by writing a simple python tool.

Note: Future work package, not in spec:

```
In order to auto-assign place types 1-6 (see dictionaries section
below), the closest place of that type to the centroid of the dataset will be
used. If no region type and region are specified at point of ingestion, the
product shall use default values of 'Local area - nearest named place'
(place_type 7 as listed in the next section) for place_type and are assigned
the named place that is nearest to the centroid of the product geometry (based
on geonames dataset lodged in the **place** dictionary), unless they have been
manually assigned.
```

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.

Note for Alessandro: use one of the django tagging apps for this.

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–
TY	2 character abbreviation e.g. RA for raster
PLACENAME	First 10 chars of place name with spaces/whitespace removed e.g. CAPETOWN–
yymmdd	temporal extent start date
yymmdd	temporal extent end date
LLLL	processing Level code eg. L4Aa; L4Ab
EPSG	EPSG Code for coordinate reference system of dataset

All id text elements will be converted to upper case. This a sample GeoSpatial product ID may look like this:

```
ORD_LANDUSE---_VL_ZA--------_110101_110211_L4Ab_4326
```

The order is of the id elements is designed so that when listing products using e.g. the unix 'ls' command they naturally appear grouped thematically, then by data type and then by place.

Geospatial Product Aggregation Rules

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

Geospatial Product Dictionaries

The following dictionaries are implemented to support the creation of geospatial products.

Place Type

A place 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
8	Quarter Degree Square - using ZA convention e.g 3318BC

Administering the place types list is not an end-user activity and should be done in consultation with developers.

Place

For places, a dictionary based on GeoNames provides a near-exhaustive list of local areas and towns. The geonames table will be augmented on an adhoc manner as required via the admin panel. When the place represents an area, the centroid of that area will be used in the geometry.

The place model looks like this:

ID	Type	Name	Long Name	Geometry
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	

Note: Alessandro we need an admin ui for this

4.1.7 Ordinal Products

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

Concrete or Abstract: Concrete

For ordinal products, there are three types of accuracy defined:

1. spatial

2. temporal
3. thematic

The first two are catered for by the GeospatialProduct abstract base class (equivalent_scale, temporal_extent_start and temporal_extent_end). Thematic accuracy for a product is described by adding the fields

Ordinal Product Properties

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

- class_count
- confusion_matrix
- kappa_score

The **class count** property is a compulsory simple numeric value representing how many classes are defined in the dataset.

The **confusion_matrix** property is a collection of comma separated integers representing (in order):

1. true positive (tp)
2. false negative (fn)
3. false positive (fp)
4. true negative (tn)

Thus an entry of 10,5,2,8 signfies:

1. tp=10
2. fn=5
3. fp=2
4. tn=8

The confusion matrix entry is not mandatory.

The [**kappa score http://en.wikipedia.org/wiki/Cohen's_kappa**] is a statistical measure of inter-rater agreement for categorical items. It is represented as a single, non-compulsory real number.

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'.

Ordinal Product Aggregation Rules

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

Ordinal Product Dictionaries

No additional dictionaries are introduced by the ordinal product model.

4.1.8 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

Continuous Product Properties

For Continuous data there are three additional fields:

- range_min - floating point not null
- range_max - floating point not null
- unit - foreign key

The **range_min** is the smallest value in the continuim of data represented in the dataset. This value is required.

The **range_max** is the largest value in the continuim of data represented in the dataset. This value is required.

The **unit** property is a reference to the unit dictionary and it describes the units of measurement for the data represented in the dataset. This value is required.

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'.

Continuous Product Aggregation Rules

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

Continuous Product Dictionaries

Unit - a look up table storing measurement units:

Abbreviation	Name
m	Meters
km	Kilometers
etc.	

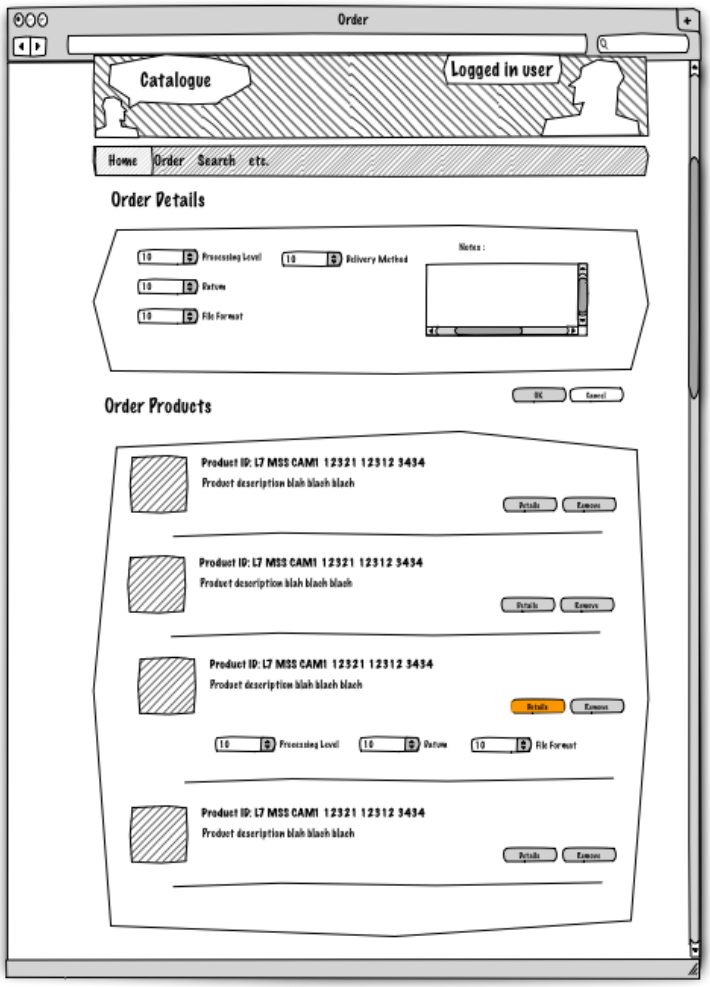
4.2 Catalogue Schema : Orders

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:

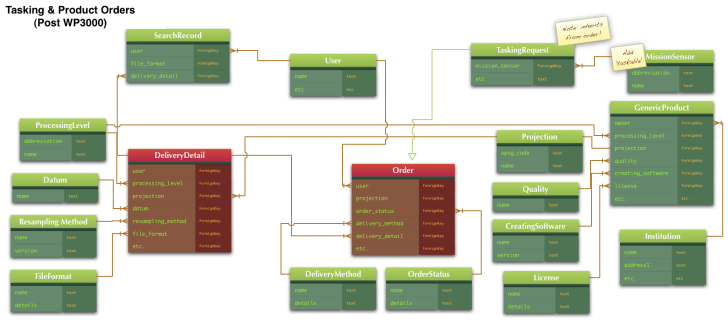
1. options that are global to the order can be defined
2. options that are specific to each product within the order can be defined (overriding the global options where applicable)

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.

4.2.1 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 Order optionally stores Order level AOI / clip mask used by staff to use to clip the products before shipping ordered products to the client
- 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.)

4.2.2 Operational notes

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

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).

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.

4.2.3 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 DIMS 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.

4.2.4 OS4EO

A Django management command, suitable to run from cron jobs, will take care of the OS4EO order placing for GenericSensorProduct.

The followin fields in SearchRecord are involved:

1. internal_order_id
2. product_ready
3. download_path

   One field in GenericSensorProduct tells us if the products is in DIMS and can be ordered with OS4EO:

4. online_storage_medium_id

   Logic is:

1. only GenericSensorProduct with online_storage_medium_id NOT NULL are DIMS products
2. if internal_order_id is not set, OS4EO order was not yet placed
3. if internal_order_id is set but download_path is NULL, the product OS4EO order was placed but not yet completed

   Pseudo code:

4. for each pending order (i.e. GenericSensorProduct items wich are not ready from orders with status that is not 'Cancelled', 'Awaiting info from client', 'Completed')
   a) if the product is ordered (internal_order_id is set) checks order status
      i. if the status is 'completed' check the file availability
         A. if the product is available,
         B. fill download_path and save the item (this triggers product_ready = True)
         C. if all other items from the same order are ready
         D. change order status to 'Completed'
   b) if the product is not ordered, order it:
      i. call the OS4EO service to place the order, fill internal_order_id with the id supplied from OS4EO

OS4EO configuration

The WS endpoint is specified in global settings.py:

```
OS4EO_SERVICE_ENDPOINT = "http://196.35.94.243/os4eo"
```

OS4EO command

Note: the command is not yet finished nor tested because of some problems that Werum should solve: http://redmine.linfiniti.com/issues/21

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

```
$ python manage.py os4eo_order -h
Usage: manage.py os4eo_order [options]

Place orders

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
  -t, --test_only       Just test, nothing will be written into the DB.
  --version             show program's version number and exit
  -h, --help            show this help message and exit
```

OS4EO client library

The client library is located in catalogue/os4eo_client.py, tests and example usage are in catalogue/tests/os4eo_client_test.py.

The following methods have been implemented so far:

1. GetCapabilities
2. Submit
3. GetStatus
4. DescribeResultAccessResponse

Note: the library development has been carried out incrementally and only the minimum set of parameters needed to successfully place and handle an order has been implemented so far. Additional parameters can be easily added in the future as needed.

Sample session:

```
>>> from catalogue.os4eo_client import OS4EOClient
>>> os4eo = OS4EOClient(debug=False)

>>> os4eo.GetCapabilities()
[<Element '{http://www.opengis.net/ows}Operation' ...

Place single order

>>> single_id, single_submit_status = os4eo.Submit( \
  ['SPOT5.HRG.L1A:/eods_hb_pl_eods_XXXXB00000000253027605285/eods_hb_pl_eods_//SPOT5.HRG.L1A'], \
  '100001', \
)

Place multiple
```

```
>>> multiple_id, multiple_submit_status = os4eo.Submit( \
  ['SPOT5.HRG.L1A:/eods_hb_pl_eods_XXXXB00000000253027605285/eods_hb_pl_eods_//SPOT5.HRG.L1A', \
  'SPOT5.HRG.L1A:/eods_hb_pl_eods_XXXXB00000000253027605866/eods_hb_pl_eods_//SPOT5.HRG.L1A' ], \
  '100002', \
)


>>> os4eo.GetStatus(single_id)

>>> os4eo.GetStatus(multiple_id)

>>> os4eo.GetStatus(single_id, True)

>>> os4eo.GetStatus(multiple_id, True)

>>> os4eo.DescribeResultAccess(single_id)

>>> os4eo.DescribeResultAccess(multiple_id)
```

Note: elementsoap library was used to develop the webservice os4eo_client library (added in REQUIREMENTS.txt) buta patched version was used instead of the official one , the patched version has enhanced debug capabilities. When debug is not needed anymore, the standard library can be used, just delete the "debug" parameter from the function calls in the main script.

4.2.5 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 zones in which the image bounding box falls, and the two zones to either side of that are shown.
- For Order records, only utm zones that intersect with the image bounding box 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

4.2.6 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.

4.2.7 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.

4.2.8 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)

4.2.9 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).

4.2.10 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

4.3 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.

4.3.1 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.

4.4 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.

4.4.1 Simple search

Simple search is the default search interface. The use can enter the following search criteria:

1. date ranges (multiple, required)
2. digitized area of interest (optional)

   A new search record is created when the user submits a valid form.

   === Advanced search ===

   When the user press "Advanced search", many more fields appear and a complex interface allows to specify almost all other search criteria, which can be different according to the selected product type.

   For sensor-based products, the sensor criteria is always mandatory and an Ajax powered method, keeps available choices consistent in the sensor-related criteria (i.e. when the user selects a certain mission, only the sensors available in that mission are shown in the sensor box).

A new search record is created when the user submits a valid form.

Note: this Ajax U is similar but not identical to what happens in the Product ID search interface: in advanced search, the Ajax call takes only care of drilling down the mission-sensor hierarchy without issuing a new search query on every change.

Product ID search

This search view is only available for optical search products because other types have no Product ID implementation.

The aim of this function is to facilitate search results narrowing of an existing advanced search results set, through a UI based on the product ID.

A link to the search view is available from the search results window only if the search is an advanced search for optical products.

Note: every search submit will not create a new search but modifiy the original one. This is not identical to the behaviour of the modify search view which always creates a new search object.

====== Ajax search implementation details ======

The Ajax product id search has a JS handler which issues an Ajax post call on every option select change.

The Ajax call is received by ‘catalogue.views.search.ProductIdSearch‘ which takes care of:

1. updating the Search object with the new parameters
2. calls catalogue.views.searcher.Searcher.describeQuery, that return a JSON object containing
   a) the query description (to update the "Search summary" box)
   b) the query SQL (if settings.DEBUG is set)
   c) a dictionary of values for the select options (e.g.: sensor_type, acquisition:mode etc.)

   The JS code responsible for this is in the view template catalogue/templates/ProductIdSearch.html and updates the select options hiding/showing the available values

   Note: in order to avoid a narrow/one-way only search, all options must be nullable, this means that also sensors multiple select was changed to be not mandatory.

   Note: catalogue.views.searcher.Searcher.describeQuery accepts an optional parameter (unset_only=False) that if activated, returns only values for parameter that are not set in current Search, this allows for a pivot-like way of obtaining the values: "tell me all the possible values for the parameters that are not set", this can be useful if the UI do not use '*' or other similar mechanisms to delete a parameter from the filter.

5 Search

6 Searching for data on the Catalogue

There are two ways that a user can carry out a search on the catalogue:

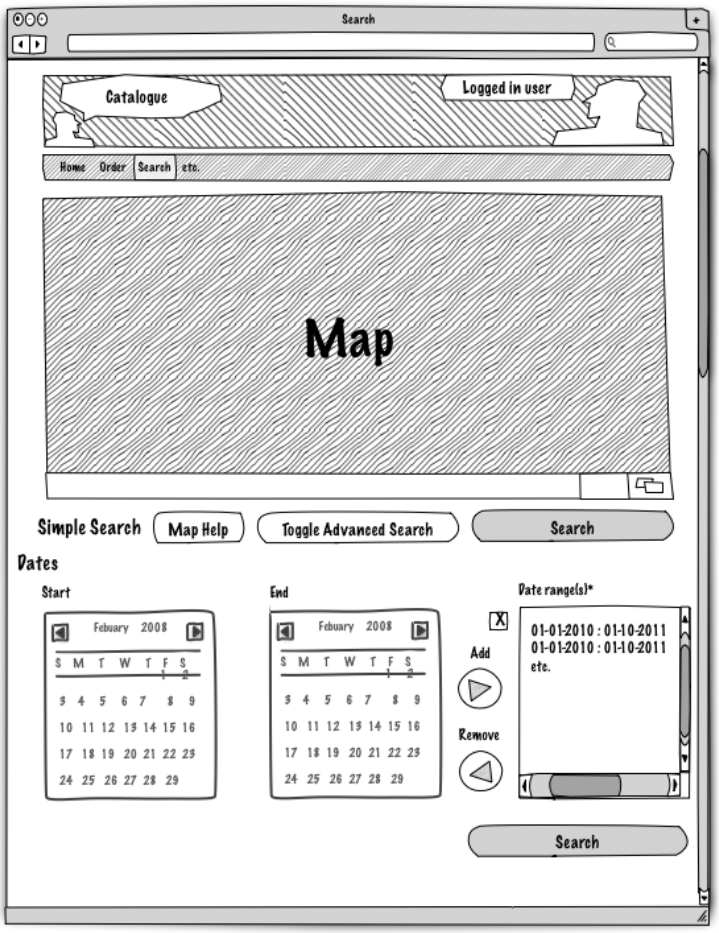
1. Basic search - for novice / casual users this is the easiest since it requires very little domain knowledge.
2. Advanced search - for power users and remote sensing professinals. Requires moderate to in-depth domain knowledge.

These two search environments are described in the sections that follow.

In addition to the advanced search, there is also the option for the user to refine searches using a 'product id refinement'. In this process, the user can further filter their search results and view the updated result set in real time.

6.1 Basic Search

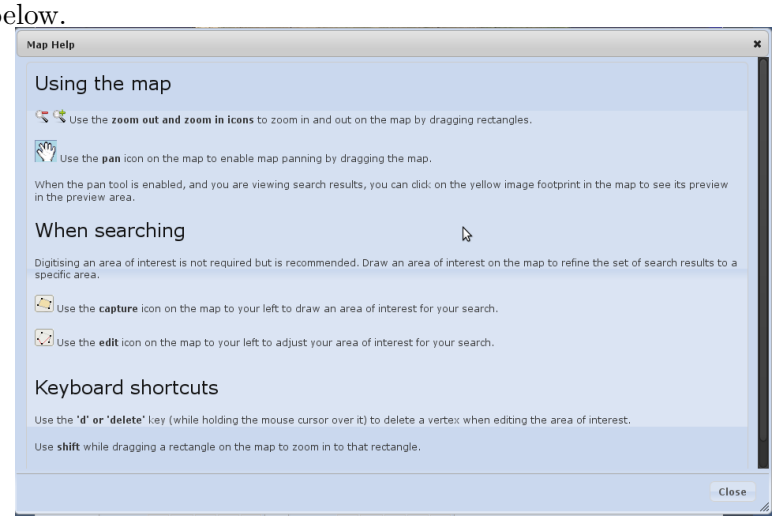
Basic search requires only the specification of a date range and optionally the digitisation of a search area on a map. A wireframe illustrating this is provided below.



At any time, the user can switch to the advanced search dialog (as described in the

next sections) by using the 'Toggle Advanced Search' button.

The map area of the search window (which remains when the search is toggled to advanced mode) is provided above the search form. A button on the search form will launch a dialog that shows useful information on using the map. This is illustrated below.



The map itself can be customised by selecting the 'Layers' tab to the right and switching it to use a different backdrop layer. The layers listed may vary based on the user rights of the person currently logged in. Users that have been assigned 'partner' status will see higher resolution SPOT5 mosaics (which are not shown to general users due to licensing restrictions).

The toolbar below the map has a number of tools which can be used to interact with the map. These allow the activities of:

- zooming in
- zooming out
- panning (shifting the map extents north, south, east or west)
- stepping back in the history of areas of extents places you have panned or zoomed to)
- stepping forward in the history of areas of extents (places you have panned or zoomed to)
- digitising an area of interest for your search
- editing the digitised area of interest
- deleting the digitised area of interest

In addition, to the bottom right of the map area, two icons are present. These allow you to:

- restore the map size to default
- grow the size of the map such that if occupies the entire browser window area

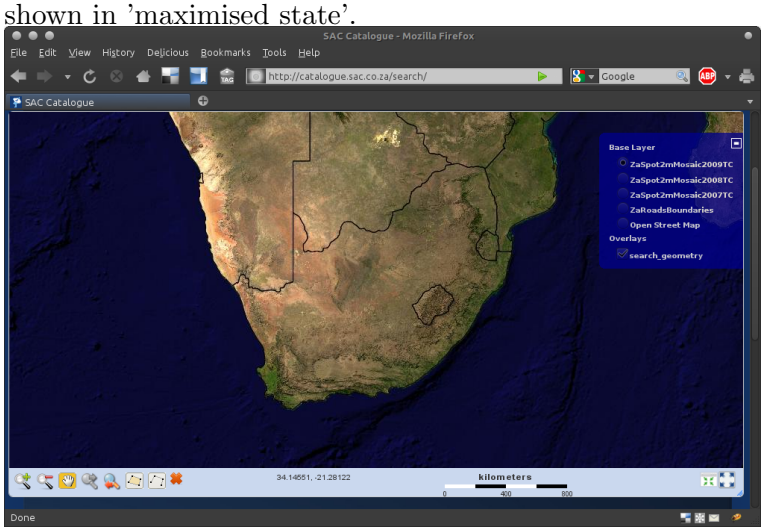
It should be noted that for the latter item, the search form and related buttons are shifted down to 'below the fold' (i.e. below the bottom edge of the window), requiring the user to scroll the window down to once again see the other input controls on the form.

Below the map are two additional elements:

- the current position of the cursor in decimal degrees
- a scale bar with units in kilometers

Technical note: The backdrop layers shown in the map areas are in the 'Google Mercator' coordinate reference system (EPSG:90013)

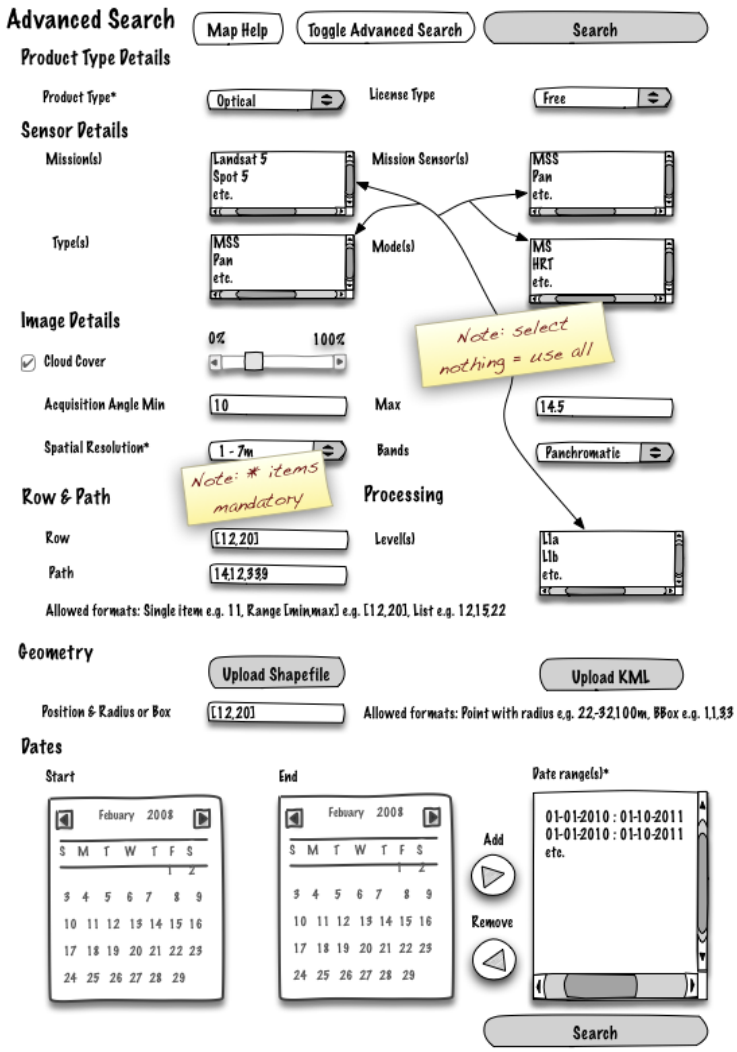
In the image below, the various map elements described above are shown. The map is shown in 'maximised state'.



6.2 Advanced Search

For casual users, the simple search functionality may be sufficient, but there is a class of users that demand more control over the search criteria. These users include SAC staff members, remote sensing specialists and other professional users. To cater to the needs of this demographic, we have implemented the advanced search functionality.

6.2.1 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.
- The Mission, Mission Sensor, Type and Mode selectors are constrained to show only non-radar products (based on MissionSensor::is_radar property).

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 - 1m
1m - 7m	1 - 7m
7m - 25m	7 - 250m
25m - 70m	25 - 70m
70m - 1km	70 - 1000m
> 1km	1000m +

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.

Note: Although the normal expectation is that both a row and a path are provided, this is a soft requirement, and a user may enter one or the other (or neither) if they desire.

Processing levels

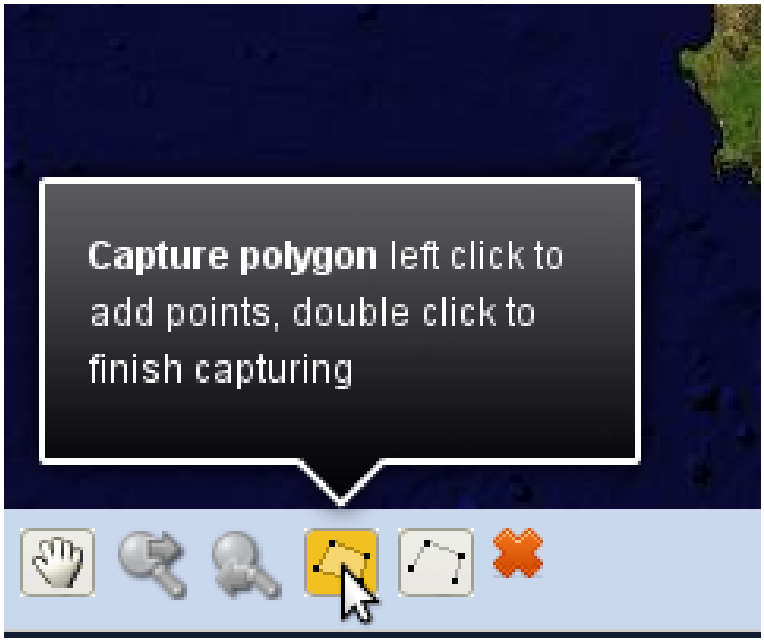
The products can be filtered according to their processing level. The list of processing level list is not dynamic (i.e. it does not ajust its entries based on mission, sensor etc. selections) as it would be prohibitively CPU intensive to do so. As such there is no guarantee of being at least one product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

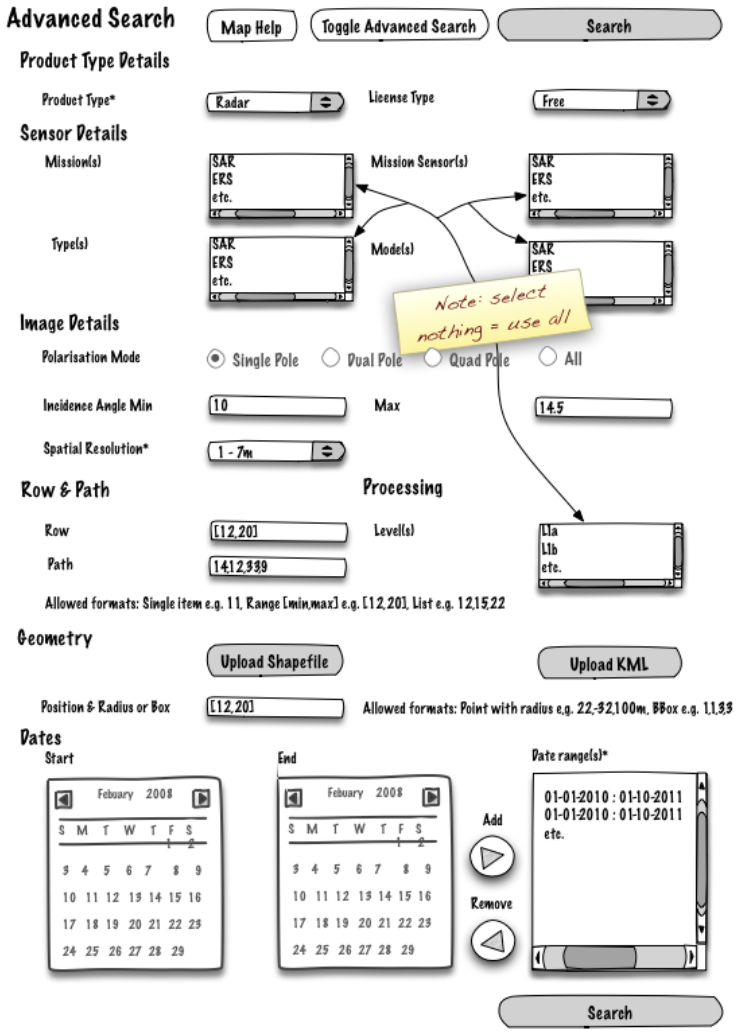
- Selecting an end date in the 'end' calendar
- Clicking the 'add' icon to add the date range to the range list

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.

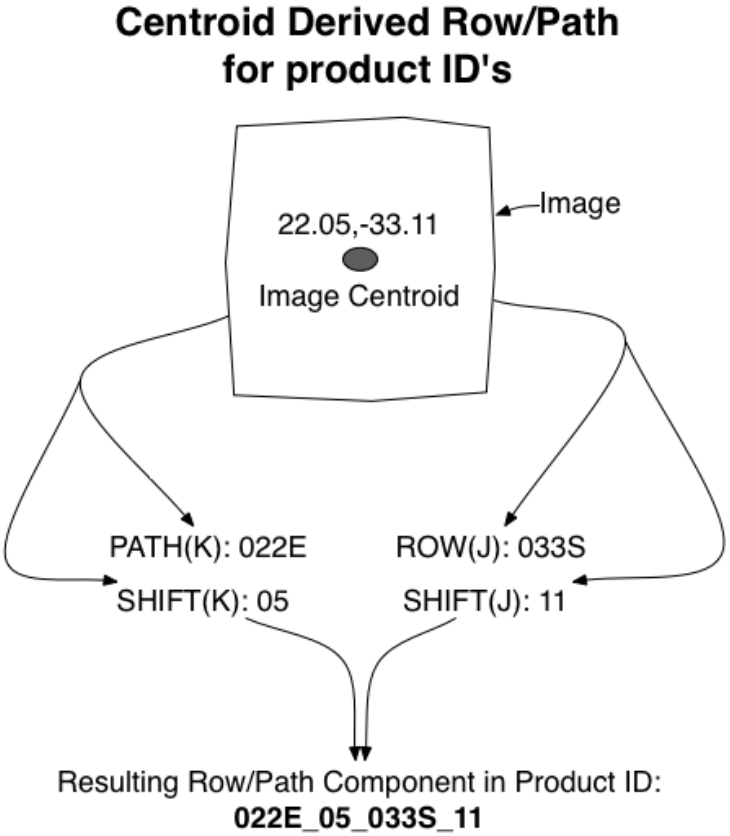
6.3 Radar Search



Radar products share many properties in common with Optical products, so you should read the description of the Advanced Search for Optical Products above first. There are five principle differences in the Radar Search dialog:

1. There is no option to choose a cloud cover rating.
2. There is an option added for polarisation mode.
3. Acquisition angle is replaced with incidence angle (RadarProduct::incidence_angle).
4. The Mission, Mission Sensor, Type and Mode selectors are constrained to show only radar products (based on MissionSensor::is_radar property).
5. There is no option to choose a number of bands.

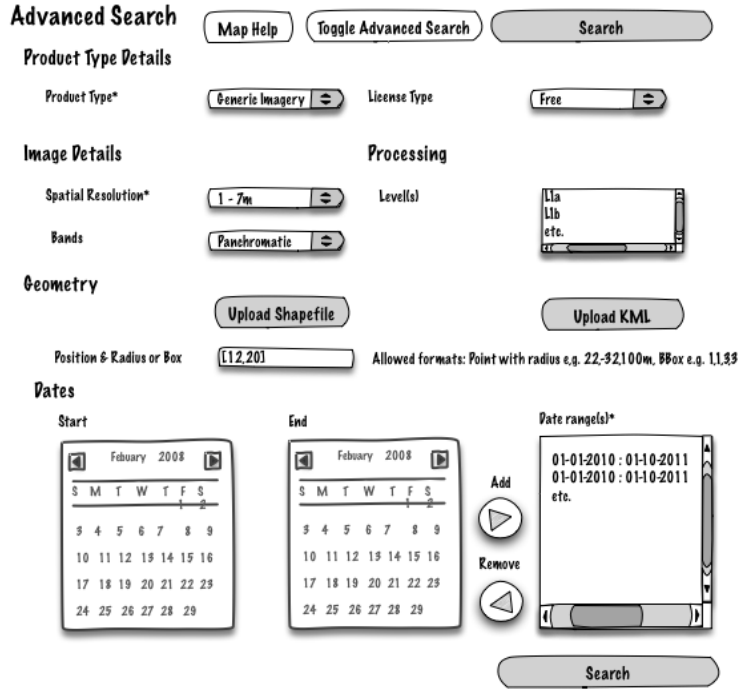
It should be noted that radar products will often not have row/path data. In this situation the row/path and offsets should have been defined using the product centroid system as illustrated below. For this reason the row/path inputs remain.



The centroid based the row/path schema uses N/S/E/W suffixes to indicate hemisphere, however the row/path range definition expects numeric entries. As such negative values in the row box will indicate west and negative values in the path will indicate south.

6.4 Generic Imagery Search

The purpose of the generic imagery search is to allow the user to locate GenericImageryProducts and their class heirarchy descendents (currently Optical and Radar products). The generic imagery search uses a reduced set of options in order to cater for the range of different imagery types.

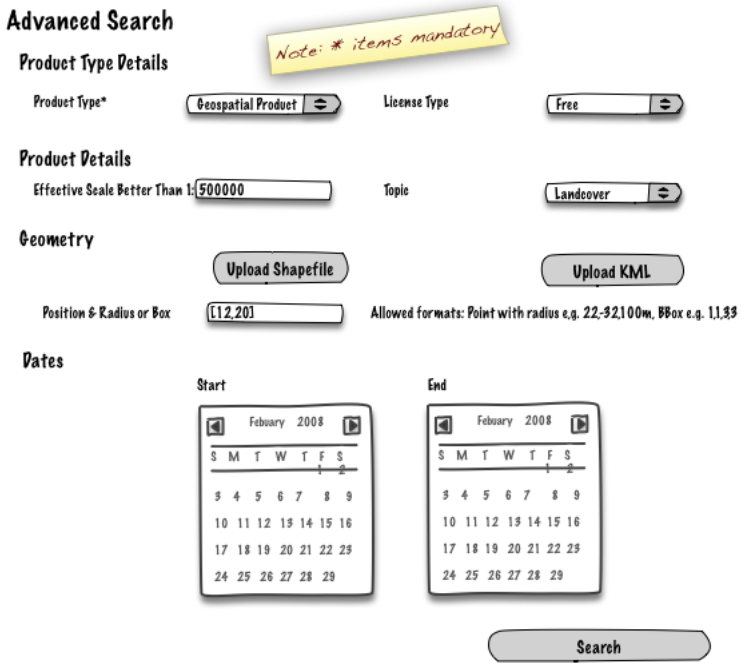


In particular:

- There are no options related to the selection of a mission, sensor, sensor type or acquisition mode.
- There are no options related to acquisition angle or incidence angle.
- There are no options relating to polarisation mode.
- There are no options relating to Row / Path.

Because of the diverse results that may be returned apon completing a generic imagery search, there is no post search product id filtering catered for. However the user may refine the search from the search results screen where they will be returned to a pre-populated instance of the original search form.

6.5 GeospatialProduct Search



7 Reporting Tools

7.1 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.

7.1.1 Order summaries

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:

Id	Date	Status	Placed by	View

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.

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

7.1.2 PDF report generation

To generate on-the-fly PDF reports we use **Pisa** http://www.xhtml2pdf.com which is an HTML/XHTML/CSS to PDF converter written in Python and based on Reportlab Toolkit, pyPDF, TechGame Networks CSS Library and HTML5lib. This apporach doesn't break normal Django application development, as we can use Django Templating engine to prepare HTML for PDF rendering.

Graphs in PDF reports are created using Google Chart API through **pygooglechart** Python library.

7.2 Heatmap implementation

Heatmap is based on a pre-calculated data which enables offline map generation for specific periods of time, i.e. last week, last month, last year, etc.

It's based on discovering intersections of user area of interest geometry for each search, with quarter degree grid for the whole world. These intersections are then summarized for each *grid cell* per day. To create actual heatmap we are using following GDAL utilities (*gdal_grid*, *gdaldem* and *gdalwrap*) and ImageMagick to combine images into meaningful map.

Heatmap data and rendering is updated by a scheduled background process, which creates required heatmaps. Heatmap visualization is currently implemented as simple OpenLayers Image layer.

7.2.1 Heatmap configuration

Following steps are required to create and configure heatmap generation functionality.

1. execute database preparation script
   - `psql -f sql/migrations/200-heatmap-reporting.sql -d sac`
   - this script cleans existing invalid geometry data, creates tables *heatmap_grid* and *heatmap_values* and database procedures *heatmap_grid_gen*, *update_heatmap* and *heatmap*
2. execute data initilization script
   - `psql -f sql/migrations/201-heatmap-bootstrap.sql -d sac`
   - this script first generates quarter degree grid by calling `SELECT heatmap_grid_gen(0.25);`
   - then updates data in *heatmap_values* table by calling `SELECT update_heatmap();` without parameters

7.2.2 Heatmap creation

To create heatmaps we use `generate_heatmaps.sh` script which is in project *scripts* folder. This script is configured to be executed relative to *scripts* folder, however, variables which control script behavior and output folders can be modified in script itself.

Script is preconfigured to create 4 heatmaps based on last week, last month, last three months and full dataset.

First we update *heatmap_values* table with new data, if new data exists. Then for each specified heatmap we use *gdal_grid* to extract point data from database for required date range and apply *invdist* spatial data approximation function, and create raster of required size. After gridding we colorize raster using *gdaldem* by using colors specified in `heatmap.txt` file. In the end we transform created raster to world mercator (EPSG:900913) projection using *gdalwrap* and overlay prepared country borders over it, using ImageMagick.

Script runtime depends on amount of data. Currently it takes around 30 minutes to generate 4 heatmaps, with most of the time spent generating full dataset heatmap.

7.3 World borders data

World borders data is essential for generating reports and heatmaps. For this purpose we are using preprocessed datasat available at http://thematicmapping.org, specifically http://thematicmapping.org/downloads/TM_WORLD_BORDERS-0.3.zip. This dataset is already in the repository `resources/world_borders` directory.

7.3.1 World borders loading

Process of loading world borders is done in three steps:

1. execute database preparation sql script
   - `psql -f sql/migrations/202-report-query-by-country.sql -d sac`
   - this script creates required database table and indexes
   - **Note:** on a new installation this step is not required, because this table is created on project initialization
2. importing world borders data to world borders table
   - `./manage.py runscript load_world_borders`
3. execute data sanitizing script
   - `psql -f sql/migrations/203-worldborders-data-sanitization.sql -d sac`
   - this script removes `'` `(` `)` characters which break PISA report generation functionality

Note: all paths are relative to project root dir

8 Ingestors

for system management and for ingesting products.

- dims-ingest
- modis-ingest
- misr-ingest
- rapideye_harvest

8.1 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

8.1.1 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.

8.1.2 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 Editition (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
```

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
```

8.1.3 Trouble shooting and general tips

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.
```

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
```

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)
```

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
```

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";
```

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.

Accessing the server Interactively

```
ssh 196.35.94.210 -l informix
```

Interactive database access:

```
dbaccess
```

8.1.4 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
```

Command line processing using echo

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

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

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
```

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
```

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!

8.1.5 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
```

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.

8.1.6 SPOT Image Data

8.1.7 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.

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
|   +-- ThN1
+-- 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.

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.

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.

8.2 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.

Note3: The most important missing piece in this import procedure is the DIMS identification id (to be used in OS4EO ordering process): this id is still missing from the ISOMetadata.xml file and hence it is not possible to import DIMS packages that can be ordered via OS4EO. The DIMS id should be available and stored into GenericSensorProduct.online_storage_medium_id. This id must be the same that we can use to usbmit an order with OS4EO ”Submit” method

8.2.1 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
```

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)

8.2.2 Implementation details

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

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.

8.3 Procedures for importing data from RapidEye 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.

8.3.1 Importing the packages from the remote catalogue

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

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

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

Imports RapidEye packages into the SAC catalogue

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
  -u USERNAME, --username=USERNAME
                        Username for HTTP Authentication. Defaults is read
                        from settings.py.
  -p PASSWORD, --password=PASSWORD
                        Password for HTTP Authentication. Defaults is read
                        from settings.py.
  -b BASE_URL, --base_url=BASE_URL
                        Base catalogue URL. Defaults is read from settings.py.
  -t, --test_only       Just test, nothing will be written into the DB.
  -o OWNER, --owner=OWNER
                        Name of the Institution package owner. Defaults to:
                        Rapideye AG.
  -s CREATING_SOFTWARE, --creating_software=CREATING_SOFTWARE
                        Name of the creating software. Defaults to: Unknown.
  -y YEAR, --year=YEAR  Year to ingest (4 digits). Defaults to: current year
  -d DAY, --day=DAY     Day to ingest (2 digits). Defaults to None
  -m MONTH, --month=MONTH
                        Month to ingest (2 digits). Defaults to: current month
  -l LICENSE, --license=LICENSE
                        Name of the license. Defaults to: SAC Commercial
                        License
```

```
  -a AREA, --area=AREA  Area of interest, images which are external to this
                        area will not be imported (WKT Polygon, SRID=4326)
  -q QUALITY, --quality=QUALITY
                        Quality code (will be created if does not exists).
                        Defaults to: Unknown
  -r PROCESSING_LEVEL, --processing_level=PROCESSING_LEVEL
                        Processing level code (will be created if does not
                        exists). Defaults to: 1B
  --version             show program's version number and exit
  -h, --help            show this help message and exit
```

Running the command:

```
$ python manage.py rapideye_harvest -v 2 -a 'POLYGON(( 22 3, 23 3, 23 2, 22 2, 22 3))' -d 12 -y 2011 -m 03
```

8.3.2 Global settings

The following parameters in settings.py control the process:

```
# RapidEye catalogue ingesting settings
CATALOGUE_RAPIDEYE_BASE_URL = 'https://delivery.rapideye.de/catalogue/'
CATALOGUE_RAPIDEYE_USERNAME = '*********'
CATALOGUE_RAPIDEYE_PASSWORD = '*********'
```

8.3.3 Implementation details

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

Pseudocode:

1. given the year and month (and optionally the day)
2. download the shapefiles with all imagery metadata for that period of time
3. optionally clip to a bounding polygon to restrict area of interest
4. for each image:
   a) calculate product_id
   b) search in catalogue
   c) if already in catalogue, process the next product
   d) download thumbnail
   e) ingest in catalogue

Data and metadata extraction

The following metadata are hardcoded constants and can be changed at the top of the script 'catalogue/management/commands/rapideye_harvest.py':

Parameter	Value
band_count	5
radiometric_resolution	16
geometric_resolution_x	5
geometric_resolution_y	5
mission_sensor	REI
sensor_type	REI
acquisition_mode	REI
projection	ORBIT
product_acquisition_start_time	09:00

The following metadata are passed to the script as option parameters:

Parameter	Default
license	SAC Commercial License
owner	Rapideye AG
processing_level	1B
creating_software	Unknown

The following metadata are read from the shapefile:

Parameter	From
original_product_id	PATH
product_acquisition_start[1]	ACQ_DATE
row/path informations	footprint centroid
mission	CRAFT_ID
cloud_cover	CCP
sensor_viewing_angle	VW_ANGLE
solar_zenith_angle	90Â° - SUNELVN
solar_azimuth_angle	SUNAZMT
sensor_inclination_angle	IND_ANGLE

[1] time taken from constant product_acquisition_start_time

8.4 Procedures for importing data from MODIS 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.

8.4.1 Importing the packages from a the remote catalogue

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

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

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

Imports mdois packages into the SAC catalogue

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
  -i, --store_image     Store the original image data extracted from the
                        package.
  -m MAXPRODUCTS, --maxproducts=MAXPRODUCTS
                        Import at most n products.
  -t, --test_only       Just test, nothing will be written into the DB.
  -s, --rcfileskip      Do not read or write the run control file.
  -o OWNER, --owner=OWNER
                        Name of the Institution package owner. Defaults to:
                        MODIS.
  -l LICENSE, --license=LICENSE
                        Name of the license. Defaults to: SAC Free
                        License
  -q QUALITY, --quality=QUALITY
                        Quality code (will be created if does not exists).
                        Defaults to: Unknown
  -r PROCESSING_LEVEL, --processing_level=PROCESSING_LEVEL
```

```
                      Processing level code (will be created if does not
                      exists). Defaults to: 1B
  --version           show program's version number and exit
  -h, --help          show this help message and exit
```

Running the command:

```
$ python manage.py modis_harvest
```

8.4.2 Run control

An rc file is created or updated after every single successful import.

The rc file is located in the project's main folder and named modis_harvest.rc

The rc file contains the following informations:

1. last_folder
2. last_package

Example:

```
last_folder=2000.02.18
last_package=2000.02.18/MCD43A2.A2000049.h19v12.005.2006269115820.hdf
```

8.4.3 Implementation details

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

Pseudocode:

1. get the list of dates from remote FTP
2. sort the dates
3. read from the .ini file the last date examined and the last package imported
4. seek the list to the date next to the last imported one (if any)
5. loop for all dates
    a) scan the date folder and get the list of packages
    b) sort the list
    c) seek the list to the package next to the last imported one (if any)
    d) loop for all packages
        i. download image
        ii. read metadata
        iii. calculate product_id
        iv. search in catalogue
        v. skip if already in catalogue
        vi. creates the thumbnail
        vii. ingest in catalogue

8.4.4 Data and metadata extraction

The following metadata are hardcoded constants and can be changed at the top of the script 'catalogue/management/commands/rapideye_harvest.py':

Parameter	Value
band_count	5
radiometric_resolution	16
geometric_resolution_x	500
geometric_resolution_y	500
mission	maps Terra/Aqua to MYD and MOD
mission_sensor	maps Terra/Aqua to MYD and MOD
sensor_type	MOD
acquisition_mode	MOD
projection	ORBIT

The following metadata are passed to the script as option parameters:

Parameter	Default
license	SAC Commercial License
owner	MODIS
processing_level	1B

The following metadata are read from the metadata in the HDF image:

Parameter	From
original_product_id	LOCALGRANULEID
product_acquisition_start	RANGEBEGINNINGDATE RANGEBEGINNINGTIME
product_acquisition_end	RANGEENDINGDATE RANGEENDINGTIME
row/path informations	footprint centroid
mission	CRAFT_ID
creating_software	HDFEOSVersion

8.4.5 Thumbnail image generation

The thumbnail is built using external gdal calls in three steps:

1. gdal_translate extracts sub dataset from hdf image (creates 4 tiffs, one for each sub dataset)
2. gdal_merge.py merge bands 1 4 3 into RGB tiff
3. gdal_translate creates a JPEG 400x400 px thumbnail with a world file

8.4.6 External programs required

1. gdal_translate
2. gdal_merge
3. bzip2

8.5 Procedures for importing data from MISR 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.

8.5.1 Importing the packages from a local folder

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

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

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

Imports MISR packages into the SAC catalogue

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
  -b BASE_PATH, --base_path=BASE_PATH
                        Base catalogue path. Default is read from settings.py.
  -i, --store_image     Store the original image data extracted from the
                        package.
  -m MAXPRODUCTS, --maxproducts=MAXPRODUCTS
                        Import at most n products.
  -t, --test_only       Just test, nothing will be written into the DB.
  -k, --rcfileskip      Do not read or write the run control file.
  -o OWNER, --owner=OWNER
                        Name of the Institution package owner. Defaults to:
                        MISR.
  -s CREATING_SOFTWARE, --creating_software=CREATING_SOFTWARE
                        Name of the creating software. Defaults to: Unknown.
  -l LICENSE, --license=LICENSE
                        Name of the license. Defaults to: SAC Commercial
                        License
  -q QUALITY, --quality=QUALITY
                        Quality code (will be created if does not exists).
                        Defaults to: Unknown
  -r PROCESSING_LEVEL, --processing_level=PROCESSING_LEVEL
                        Processing level code (will be created if does not
                        exists). Defaults to: 1B2
  --version             show program's version number and exit
  -h, --help            show this help message and exit
```

Running the command:

```
$ python manage.py misr_ingest
```

The packages are searched in three folders :
'MB2LME.002', 'MB2LMT.002', 'MI1B2E.003'

- MB2LME MISR Level 1B2 Local Mode Ellipsoid Radiance Data Ellipsoid projected TOA parameters for the single local mode scene, resampled to WGS84 ellipsoid. HDF-EOS Grid
- MB2LMT MISR Level 1B2 Local Mode Terrain Radiance Data Terrain-projected TOA radiance for the single local mode scene, resampled at the surface and topographically corrected. HDF-EOS Grid
- MI1B2E MISR Level 1B2 Ellipsoid Data Contains the ellipsoid projected TOA radiance, resampled to WGS84 ellipsoid corrected.

Note: MI1B2E.003 import is untested

8.5.2 Global settings

The following parameters in settings.py control the process:

```
# MISR catalogue ingesting settings
MISR_ROOT= '/path/to/MISR'
```

The KML file containing the blocks footprints for the different paths is located in the 'resources' folder immediately under the project folder.

8.5.3 Implementation details

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

Pseudocode:

1. scan the three folders: + MB2LME MISR Level 1B2 Local Mode Ellipsoid Radiance Data Ellipsoid projected TOA parameters for the single local mode scene, resampled to WGS84 ellipsoid. HDF-EOS Grid + MB2LMT MISR Level 1B2 Local Mode Terrain Radiance Data Terrain-projected TOA radiance for the single local mode scene, resampled at the surface and topographically corrected. HDF-EOS Grid + MI1B2E MISR Level 1B2 Ellipsoid Data Contains the ellipsoid projected TOA radiance, resampled to WGS84 ellipsoid corrected.
2. sort the list of dated ascending + if the last scanned date for the folder is set, start from the next date
3. opens nadir AN image
4. create thumbnail for AN (will be used for all other images)
5. parse metadata to get the footprint from KML file
6. transform footprint from EPSG:9001
7. get row and path from metadata
8. creates the tarball with all other camera images

Assumptions:

Main folders are fixed:

- MB2LME MISR Level 1B2 Local Mode Ellipsoid Radiance Data Ellipsoid projected TOA parameters for the single local mode scene, resampled to WGS84 ellipsoid. HDF-EOS Grid
- MB2LMT MISR Level 1B2 Local Mode Terrain Radiance Data Terrain-projected TOA radiance for the single local mode scene, resampled at the surface and topographically corrected. HDF-EOS Grid
- MI1B2E MISR Level 1B2 Ellipsoid Data Contains the ellipsoid projected TOA radiance, resampled to WGS84 ellipsoid corrected.

Data and metadata extraction

The following metadata are hardcoded constants and can be changed at the top of the script ‘catalogue/management/commands/rapideye_harvest.py‘:

Parameter	Value
band_count	4
radiometric_resolution	16
mission_sensor	REI
sensor_type	REI
projection	ORBIT

Note: Real projection is SOM (Space Oblique Mercator: http://en.wikipedia.org/wiki/Space-oblique_Mercator_projection)

The following metadata are passed to the script as option parameters:

Parameter	Default
license	SAC Commercial License
owner	MISR
processing_level	1B2
creating_software	Unknown

The following metadata are read from the imagery metadata of the nadir image:

Parameter	From
geometric_resolution_x	Block_size.resolution_x
geometric_resolution_y	Block_size.resolution_y
path	Path_number
acquisition_mode	Cam_mode
metadata	metadata (the whole)

The following metadata are read from imagery core metadata of the nadir image:

Parameter	From
product_acquisition_end	RANGEENDINGDATE + RANGEENDINGTIME
product_acquisition_start	RANGEBEGINNINGDATE + RANGEBEGINNINGTIME|
row	ORBITNUMBER

Thumbnail creation

The thumbnail is created extracting the blocks containing valid informations when in local mode. Those block number are stored in metadata as "Start_block" and "End block" (note the missing undescore in the lattest).

From the extracted blocks, individual GeoTiff georeferenced images are created, georeferencing is done through the a.m. KML path/blocks shape file.

BGR bands come from the first three band of the block image.

The three images are then resampled and merged with gdal_merge.py, there is still an unresolved alignment problem in this procedure.

8.6 Procedures for importing data from Terrasar-x 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.

8.6.1 Importing the packages from a local folder

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

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

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

Imports Terrasar packages into the SAC catalogue

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
  -b INDEX_URL, --index_url=INDEX_URL
                      Base catalogue URL. Defaults is read from settings.py.
  -t, --test_only     Just test, nothing will be written into the DB.
  -o OWNER, --owner=OWNER
                      Name of the Institution package owner. Defaults to:
                      Infoterra.
  -s CREATING_SOFTWARE, --creating_software=CREATING_SOFTWARE
                      Name of the creating software. Defaults to: Unknown.
  -l LICENSE, --license=LICENSE
                      Name of the license. Defaults to: SAC Commercial
                      License
  -a AREA, --area=AREA  Area of interest, images which are external to this
                      area will not be imported (WKT Polygon, SRID=4326)
  -q QUALITY, --quality=QUALITY
                      Quality code (will be created if does not exists).
                      Defaults to: Unknown
  -r PROCESSING_LEVEL, --processing_level=PROCESSING_LEVEL
                      Processing level code (will be created if does not
                      exists). Defaults to: 1B
  -m MAXPRODUCTS, --maxproducts=MAXPRODUCTS
                      Import at most n products.
  -f, --force_update  Force an update for exists products, default behavior
                      is to skip exists products.
  --version           show program's version number and exit
  -h, --help          show this help message and exit
```

Running the command:

```
python manage.py terrasar_harvest -m 16 -v 2 -b 'http://localhost/terrasar.zip' -a 'POLYGON(( 12 3, 23 3, 23 2, 12 2, 12 3))' -f
```

8.6.2 Global settings

The following parameters in settings.py control the process:

```
CATALOGUE_TERRASAR_SHP_ZIP_URL = 'http://terrasar-x-archive.infoterra.de/archive/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=archive:archive&outpu
```

8.6.3 Implementation details

Data and metadata extraction

The following metadata are hardcoded constants and can be changed at the top of the script 'catalogue/management/commands/terrasar_harvest.py':

Parameter	Value
band_count	5
radiometric_resolution	16
mission_sensor	SAR
mission	TSX
projection	ORBIT

The following metadata are passed to the script as option parameters:

Parameter	Default
license	SAC Commercial License
owner	Infoterra
processing_level	1B2
creating_software	Unknown

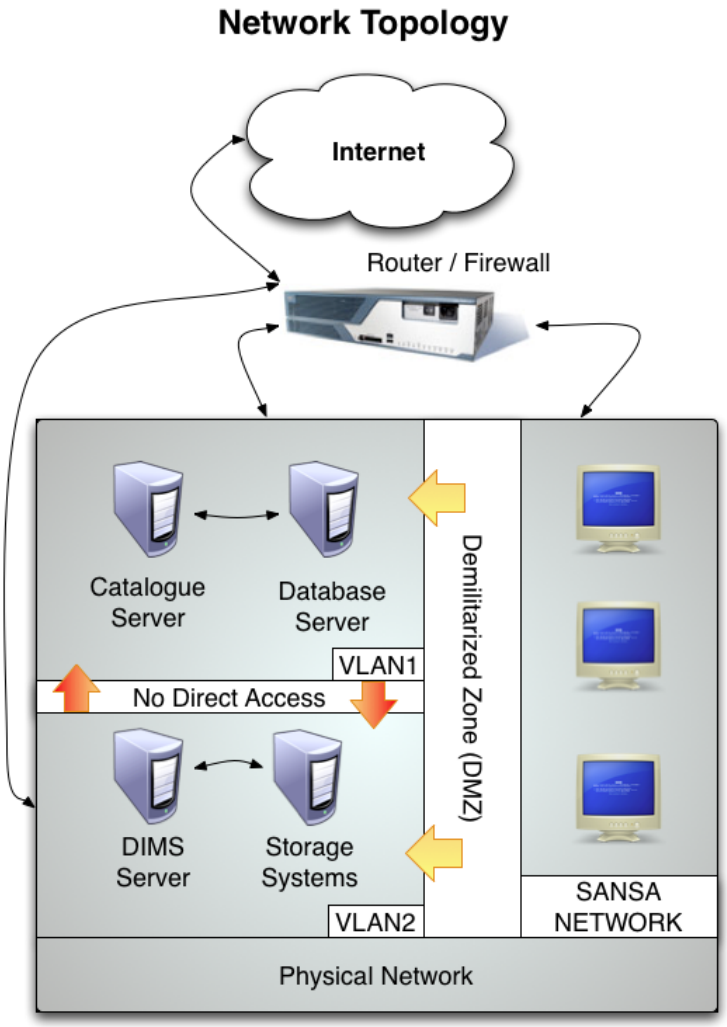
The following metadata are read from the metadata or the footprint:

Parameter	From
geometric_resolution_x	resolution [1]
geometric_resolution_y	resolution [1]
path	from footprint centroid
row	from footprint centroid
orbit	rel_orbit
polarising_mode	pol_mod
imaging_mode	img_mod
sensor_type	img_mod, pol_mod
acquisition_mode	pol_chan [2]
incidence_angle	avg(inc_min, inc_max)
metadata	metadata (the whole)

[1] the resolution field is a string field like: "Approximated Resolution Range: 3,5 - 3,3 m", the values are extracted and an average calculated. [2] Slashes are stripped.

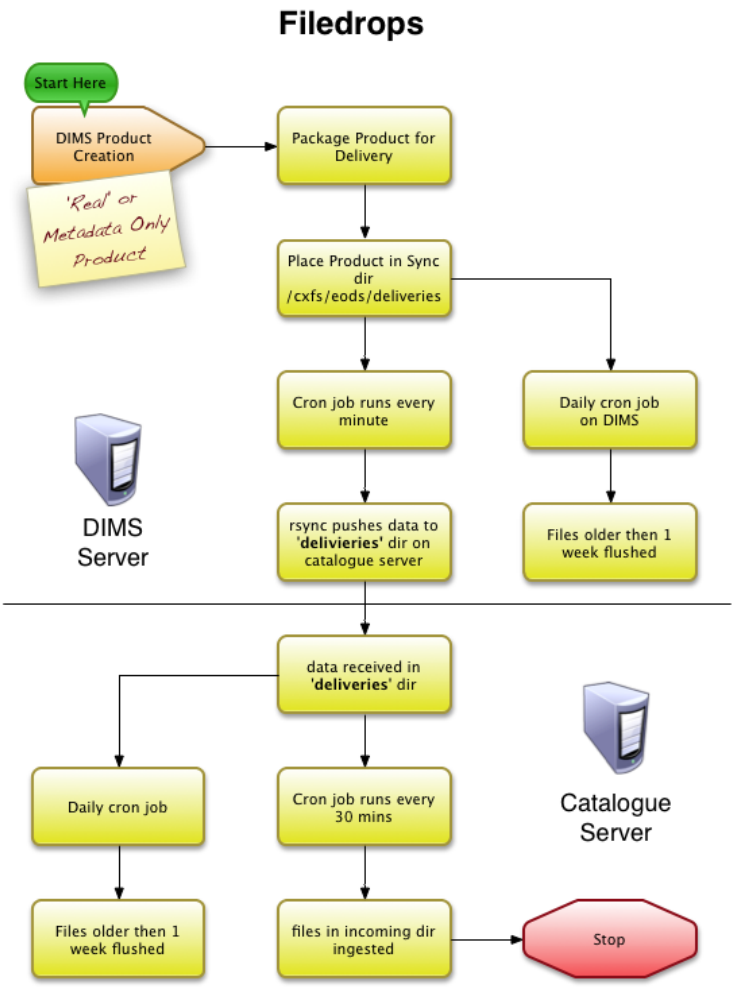
8.7 Lion File Drop

Lion (the catalogue server) should be isolated in the network from other SANSA servers and infrastructure (with the exception of the elephant database server). In addition the server has been locked down as much as possible and we do not wish to run services such as ftp on it. Thus in order to make files available to the catalogue server for processing, data should be **pushed** to the server and not **pulled** from other servers on the network. This will allow the system to be configured without making holes through the firewall to facilititate file transfers off internal servers.



To achieve a drop directory capability, the installation on the catalogue server provides rsync (remote synchronisation) access over ssh (secure shell) using rssh (restricted shell).

The following diagram illustrates the logic flow for the filedrop.



8.7.1 Setup rssh

```
sudo apt-get install rssh
```

8.7.2 Creating a filedrop user

Next we make a user with a rssh (restricted shell) that can only use scp and rsync.

```
sudo useradd -m -d /mnt/cataloguestorage2/filedrop -s /usr/bin/rssh filedrop
sudo su - filedrop -s /bin/bash
ssh-keygen -N ''
```

The -N option should suppress prompting for a passphrase - no passphrase is entered so that keybased authentication can be used during non-interactive sessions (such as from a cron job) to transfer files.

After this, a public and private keypair exist in /mnt/cataloguestorage2/filedrop/.ssh. The private key should be copied to any server that will be synchronising files into the filedrop directory. This is explained in more detail further on in this document.

Now exit the filedrop user's shell:

```
exit
```

8.7.3 Ssh configuration

Next sshd has to be configured to allow connections for the filedrop user. To do this, edit /etc/ssh/sshd_config and add the filedrop user to the AllowUsers clause:

```
AllowUsers timlinux cstephens modisdbc0 filedrop
```

And then restart sshd:

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

8.7.4 Authorized Keys for filedrop

Next we configure the filedrop user's authorised keys file so that key based connections from a specific host can be made.

Note: the 'specific host' part above is an additional security measure. This step needs to be repeated for each client that may want to deposit files into the filedrop.

```
sudo su - filedrop -s /bin/bash
cd .ssh/
cp id_rsa.pub authorized_keys
chmod 600 authorized_keys
```

Now edit authorized_keys and prepend the server ip address that will be copying files over. We will use cheetah for the purposes of this document.

```
#
# Restrict access to cheetah. Lines wrapped here for convenience in this document
#
from="cheetah",no-port-forwarding,no-pty ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEA0iDxnisywfqqzNN2CUN2xJBIOhyvoAA9uqHDagN620UFH4A4Egdg
5am0CsDT8jm1VO/Y2ZCuyKNCmkRCnhPv4IeblsHFc2ekfZsZpSPYYAupUo43POhfwcAPUvdoec1fuBJd
o+Y/zNBz8TOmAr3Mbc0zf5pLgdA3VE44TauCEt6KJO0AzqzlYEI1tmKFZ4VacgeDhEv9246HbmpEiAoW
waMlkiIWKhV1j3w1OXiMp20pSbenGnw/2dN3avWFte3Wm4DFtnAR9MwppQ+4oyGVsG6rWgmIRVfamX1p
4FeWqnOPYfe9dCIk298GgiIHpmsGHf6Ce7uKYG F7aYW0enIFEw== filedrop@lion
```

Note: You can allow additional hosts by comma separating them e.g. from="*.foo.co.za,!bar.co.za". The wildcard means all hosts from that prefix. The ! means deny access to that host.

Now exit the filedrop shell again:

```
exit
```

8.7.5 Configuring rssh

The final thing to do on the server side is to configure rssh.

```
sudo vim /etc/rssh.conf
```

Simply uncomment the rsync and scp lines:

```
allowscp
#allowsftp
#allowcvs
#allowrdist
allowrsync
#allowsvnserve
```

There is no need to restart ssh, the changes should take effect immediately on saving the file.

8.7.6 Testing from a client

To test from the client (cheetah in this case), you need to copy the **private** key over to that server

```
sudo su - filedrop -s /bin/bash
cd .ssh
scp id_rsa timlinux@cheetah:/tmp/
```

Immediately, log in to cheetah and move the key into root's home:

```
ssh cheetah
sudo su -
cd .ssh
mv /tmp/id_rsa .ssh/filedrop_private_key
```

Now do a simple test to see if you can copy over a file to the remote system:

```
touch /tmp/test222
scp -i ~/.ssh/filedrop_private_key -P 8697 /tmp/test222 filedrop@196.35.94.243:/tmp/
```

Note the -i parameter explicitly defines which private key to use for key based authentication.

The file should successfully copy over. As an additional test, verify that no ssh access is allowed for the filedrop user on lion:

```
cheetah:~ # ssh filedrop@196.35.94.243 -p 8697 -i .ssh/filedrop_private_key
                  |\_
Daniel C. Au     -' | `.
               /7      `-._
              /            `-.__________\|\||___
              \-'_                      -- |||||,`-.
                -- `-.              /7   ||||||/-  |.
                     |\            /    |||||||/`   \
                     | \  \_______..\-' |||||||||_______________
                     |  \  \         -/||||||||\                `-.
                     |  |\  \         /||||||\           /      \
                     _/  / _|  ) _______///|||\|      ______\      |\_
                     /,__/ /,__/ /,__________/ `-..___/,______________-\`=_ __--.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% ``---'

This account is restricted by rssh.
Allowed commands: scp rsync
If you believe this is in error, please contact your system administrator.
Connection to 196.35.94.243 closed.
```

Note: for additional security, the filedrop directory could be placed in a chroot environment see for example [this article http://www.cyberciti.biz/tips/howto-linux-unix-rssh-chroot-jail-setup.html]. However this was not implemented in this work package and will remain an activity for a future work package.

8.7.7 Streamlining ssh parameters on client

As you can see from above, there are a number of options that need to be passed when making the connection:

Flag	Meaning
-p 8697	Connect on the 8697 port (non standard for security reasons)
-i .ssh/filedrop_private_key	Use a specified private key
196.35.94.243	The ip address of the host to connect to
filedrop@	The user to connect as

We can automate these items by placing the following into the ~/.ssh/config of the root user on the client system.

```
 Host lion
  User filedrop
  Port 8697
  HostName 196.35.94.243
  FallBackToRsh no
  Compression yes
  CompressionLevel 9
  IdentityFile /root/.ssh/filedrop_private_key
```

With the above config file, the syntax for copying a file over is much more streamlined:

```
scp /tmp/test222 lion:/tmp/
```

8.7.8 Synchronising with a cron job

The DIMS product library extraction places products in /cxfs/eods/deliveries/ when they are ready. We can thus synchronise these over to the catalogue server as they are created and ingest their metadata.

```
echo "The filedrop directory was created by Tim for " > filedrop-README.txt
echo "synchronising files over to lion:/mnt/cataloguestorage2/filedrop" >> filedrop-README.txt
echo "Take a look at filedrop user's crontab for details on synchronisation frequency." >> filedrop-README.txt
cd filedrop
touch test
rsync -ave ssh . lion:/mnt/cataloguestorage2/filedrop/
```

Assuming that works correctly, we create a simple script that we will run from cron that performs a sync every 5 minutes (script saved as /usr/local/bin/sac/sync_dims_deliveries).

Note: The second part of this script is commented out. If enabled it will flush any files that are older then 7 days. However I do **not** recommend running that as root, so it is disabled.

```
#!/bin/bash
# This should be run in a cron job at 5 min intervals (or similar) by root to
# push dims deliveries over from Cheetah cxfs to LION
# Root needs to have rssh client key for lion configured - see
# Catalogue documentation for details.
#
# You should add an entry like this to root's crontab to run this at intervals
# you prefer.
# e.g.
# Job will run each minute
# */1 * * * * /usr/local/bin/sac/sync_dims_deliveries
#
#Add a lock file to /tmp that indicates if a sync is in progress, if it is
#exit so that we dont try to run two concurrent rsyncs.
LOCKFILE=/tmp/dims-deliveries-sync-to-lion.lock
if [ -f "$LOCKFILE" ] #skip if exists
then
  echo "Skipping (already processing): $LOCKFILE"
else
  touch $LOCKFILE
  rsync -ave ssh /cxfs/eods/deliveries lion:/mnt/cataloguestorage2/filedrop/
  rm $LOCKFILE
fi
# be very very careful if you change this / enable this
# find /cxfs/datapickup/filedrop -name "*" -mtime +8 -exec rm -f {} \;
```

Finally add a line to the crontab:

```
sudo crontab -e
```

And add this line:

```
# Job will run at 1 minute intervalse to sync dims deliveries to lion
*/1 * * * * /usr/local/bin/sac/sync_dims_deliveries
```

Lastly, if you intend to make deliveries directly downloadable, you need to share the delivery directory on the catalogue server. The configuration disallows directory listings so that users cannot download products destined for other users. Adding the following to 000-defult apache site makes the deliveries directory listable:

```
  Alias /deliveries /mnt/cataloguestorage2/filedrop/deliveries
  <Directory /mnt/cataloguestorage2/filedrop/deliveries>
    # for debugging only, show directory listings
    # Options Indexes FollowSymLinks MultiViews
    # in production, hide directory listings
    Options -Indexes FollowSymLinks MultiViews

    AllowOverride None
    Order allow,deny
    allow from all
  </Directory>
```

8.7.9 Lion Cron Jobs

On the lion server side, a similar cron job is run to:

- ingest any new packages that have arrived
- clear away any filedrop files older than 7 days

The cron job should not be run as root in order to avoid any potential side effects that may occur from broken scripts etc.

```
# be very very careful if you change this!
# avoid the .ssh dir and other user profile stuff
find /mnt/cataloguestorage2/filedrop -name "*" -mtime +8 -exec rm -f {} \;
```

9 Future workplan suggestions

catalogue in order to allow it to store a variety of different types of products, covering optical, radar, generic imagery, ordinal and continuous products. Much of this is infrastructure only at this stage without any supporting data and so could not be rigourously tested and probably wont cater for all useage scenarios. Also there were many small items we thought should be implemented but were beyond the scope of this work package. The items listed below are a selection of future enhancements that we felt would be benificial to the catalogue:

9.0.10 Automatic creation of virtual products with back referencing

Use case: A pan sharpened product is actually a cross product of (in the case of SPOT5) and 'J' image and a 'T' image so the product entry for the pan sharpened image 'JT' should store references to the images from which it was assembled When a J and T from the same pass are added to the catalogue, a 'virtual' JT product should automatically be created

Implementation plan: Establish rules for creation of composite products and automatically create these composite products for when new product records are ingested.

9.0.11 Composite product discovery / drill down

Use case: User performs a search and obtains a composite product (e.g. PanSharpened JT or Mosaicked image) as a result. In the results table an icon shows that it is a composite product and that has sub products. Clicking an icon will expand the product into a tree view containing 1 or more heirachies of nested products, each with a thumb, metadata link, add to basket icon etc.

9.0.12 Spatial search options

**Use case:* Allow user to do spatial search based on: -Location based on their ip address -Named place (using GNIS / GEONAMES gazetteer dataset) -Saved place (a collection of sptial bookmarks) -

9.0.13 Geospatial Products

The infrastructure exists in the database for storing geospatial products but no data yet exists. We can make this process 'smart' by creating ingestion routines for various common formats e.g. shapefile, mif, tif etc. Using the geonames database we can identify the nearest named place to each dataset and auto create a thematic name and other metadata based on the contents of the dataset. This would allow data to be ingested with the minimum of user interaction. This ingestion could be further automated by extending the metadata editor tool to directly 'push' metadata (and dataset thumbnail) from the desktop.

9.0.14 KMZ Improvements

It is possible to embed imagery into kmz documents. We propos an enhancement to embed georeferenced thumbnails for products in search results / cart / orders so that they appear as low resolution overlays in google earth.

9.0.15 ACS Migrator Updates

The migrator to import data and thumbs from the legacy ACS catalogue was not updated due to being out of scope and due to time constraints. This should be updated and overhauled so that it runs as a django management command (as with the newly written migrators for rapideye etc.) and can easily be invoked from a cron job.

9.0.16 Metadata XML schema review

The SANSA-EO ISO19115 profile needs to be reviewed. This should be considered a living document and should be reviewed periodically. However, there is a pressing need for a review in the short terms so that the concepts decscribed in the profile closely match the catalogue and that we can reliably ingest the metadata for any supported product type into the catalogue using only an ISO Metadata document. Currently the catalogue only ingests documents generated by DIMS. A parallel effort should be made by Werum to extend the ISO document their system produces to cater for all product types and metadata elements.

9.0.17 Catalogue Product Filtering

The new implementation of the catalogue supports filtering of search results to further reduce the number of records displayed. When a filter is applied, a new search history is created. We would like to extend this implementation so that filtering is ephemeral and that only the initial search result is saved.

9.0.18 Multi-spectral and spatial resolution support

We discovered late in the implementation phase instances where products do not model well into our schema. In particular products having multiple spectral and spatial resolutions (e.g. for modis where bands can be 8 bit, 16 bit etc). We propose a future work activity to implement support for each product being able to have 1 or more spectral resolutions (bit depths) and spatial resolutions (pixel size). In tandem existing data would be migrated and search implementation would need to be updated to cater for this.

9.0.19 Enhance security for filedrop

The use of a filedrop for depositing e.g. DIMS products onto the catalogue server for ingestion introduces some security worries. We have performed the implementation in a fairly secure manner by allowing the filedrop connection only a restricted shell. However

for improved security, we propose that further work is carried out to implement a 'chroot jail' for this file drop user.

9.0.20 User preferences

We would like to add to the user interface of the online catalogue a user preferences area where things like ordering defaults can be set, options on notifications behaviour for orders (notify when all products in order are ready, notify on product by product basis) etc.

9.0.21 OGC Ordering Service Options

Ordering Options: We would like to be able to support the propogation of ordering options (resampling method, processing level etc.) to DIMS via the OGC OS4EO interface. This would require a parallel implementation from Werum to implement it on their side too.

GetQuotation: In addition we suggest that if DIMS inplemented the GetQuotation method of the OGC EO Ordering service, we could provide catalogue users with instant pricing for non free products available in the DIMS Product Library.

Remove all locally stored products: Currently the ˜26TB of online storage on the catalogue server is used to store online copies of many imagery products. We would like the catalogue to store metadata and thumbnails only, and to dynamically fetch products from the DIMS product library when a user requests them. This should happen via the OGC OS4EO interface and the fetched products should only be stored locally on the catalogue server for 7 days before being flushed.

CSW Client to DIMS: Currently there is no reasonable way to query DIMS about its available products or to harvest metadata from DIMS *en masse*. If DIMS activate their CSW interface we will be able to automatically harvest and query the upstream DIMS product library.

In general the OGC interface DIMS could not be implemented fully due to issues with product ID's. As such the work needs to be deferred to a future work package.

9.0.22 Modularise Ingestors

Something that was illustrated to us during this work package is the future shift from relying on one or two ingestors (e.g. from ACS, Sumbandilasat) to a diverse range of ingestors that cater for all the different product types that can now be catalogued. The current ingestors have been implemented as django command modules or as python scripts run interactively on the server. We would like to change this approach to a more user friendly and modular approach where we implement a plugin interface with a well described API. Plugins would then be created and installed into the catalogue (via a simple zip file upload for administrator users). The plugin install process would be a 'point and click' operation via the web user interface and the web administrator could set options for each plugin via a simple we gui. This approach has several benifits:

- SANSA-EO can easily task vendors or other subcontractors to provide ingestor plugins when new products are to be made available in the catalogue
- SANSA-EO can share / sell / provide a 'bare bones' version of the catalogue to third parties (e.g. other ground stations) and provide only the needed ingestors for that site.
- The import process will be standardised and SANSA-EO technical staff could create novel ingestors (e.g. that get fired off at the end of a processing chain).
- The maintainability of the system will be improved as the system will be highly modularised.

9.0.23 Desktop Search Support

It would be convenient for GIS professionals to be able to search for imagery from within their desktop GIS environment. INPE has such a tool for QGIS and it provides an opportunity to evaluate search results within the context of your own data. We propose to devlop a puplic, programmatic API to the catalogue (e.g. a REST interface) and to implement QGIS & ArcGIS plugins for performing searches and visualising results from within the desktop GIS environment.

9.0.24 User Notifications

Enable notifications when the number of search results for a user's saved searches changes. This will be implemented by adding a small icon next to each row in the search results table for the user. When clicked, the icon will update to indicate that notifications are now enabled. The notifications will be sent out once a night.

9.0.25 Smart Ordering

The ordering process can be made more userfriendly and encapsulate better the logic of the application domain:

- Implement 'smart' ordering options based on processing level and product type. e.g. L1Aa products can not be delivered in tif etc.
- Smart filtering of processing levels available for ordering products. Implement logic such that only viable processing levels are listed when ordering a product.
- Smart selection of resampling algorithm for ordering products.
- Implement logic such that only viable resampling algorithms are listed when ordering a product.
- Implement per item (search record) delivery status for orders, and logic to determine when an order is complete based on all items being ready

In addition we suggest that there is an update to the ordering and acquisition programming forms so that they have similar look to advanced search forms.