# trollduction Documentation

***Release v0.2.0***

**Panu Lahtinen, Joonas Karjalainen, Martin Raspaud**

June 24, 2015

Contents

Trollduction is a configurable framework for satellite image batch production.

This documentation is a work in progress, but the most important bits should be present. The missing details will be added once noticed.

**Contents**

CHAPTER 1

# How does Trollduction work?

Trollduction builds on the principle that satellite image batch production is event based, and that different processing steps are chained together to produce the final products. This is why trollduction is a collection of independent elements of this chain, which are then communicating together through lightweight network messages.

**The different elements provided are:**

- trollstalker: triggers an event message each time a file is put in given directory
- l2processor: generates rgb/channel images when an appropriate event message is received
- gatherer: gathers polar satellite granules together given an area of interest, and sends an event message when a matching group of granules has been gathered
- aapp_runner: runs the AAPP software on raw hrpt files, when such an event message is received, to generate level 1 data
- viirs_dr_runner: runs the CSPP software on Suomi-NPP RDR files to generate level 1 data
- modis_dr_runner: runs the SPA software on EOS-Terra/Aqua PDS files to generate level 1 data
- pps_runner: triggers processing of PPS main script once AAPP or CSPP is ready with a level-1 file

CHAPTER 2

# Setting things up cheat sheet

1. **Install other required packages:**
    - libnetcdf-dev
    - libhdf5-dev
    - python-numpy
    - python-zmq
    - python-pyinotify

   and the Pytroll packages:
    - mpop - select *pre-master* branch
    - mipp - select *pre-master* branch
    - pyresample - select *pre-master* branch
    - posttroll - select *develop* branch
    - pyorbital - select *develop* branch
    - trollsift - select *master* branch
    - trollduction - select *master* branch
    - pytroll-schedule - select *develop* branch
    - pyspectral - select *pre-master* branch
    - pykdtree - select *master* branch
    - python-geotiepoints - select *develop* branch
    - trollimage - select *develop* branch
    - pycoast - select *pre-master* branch
2. **Configure mpop**
    - modify *mpop.cfg* to suit your needs
    - add configurations for satellites you are going to use (see the templates in *mpop/etc/*)
3. **Create Trollduction configuration files**
    - use *examples/trollstalker_config.ini_template* as a template
    - use *examples/l2processor_config.ini_template* as a template

- save config file to your chosen place without the *_template* ending

4. **Create Trollduction product configuration file**
    - use *examples/product_config_hrpt.xml_template* as a template
    - save the file to the path defined in your *l2processor_config.ini* without the *_template* ending
    - define your output_dir in the xml file
    - topic from the *trollstalker_config.ini* must be included in the list of topics in *l2processor_config.ini*
5. **Create logging configurations for *trollduction* and *trollstalker***
    - use *examples/l2processor_logging.ini_template* and *examples/trollstalker_logging.ini_template* as templates
    - check the log filename (by default logs go to */tmp/* directory)
    - save these configs to the path defined in your *l2processor_config.ini* and *trollstalker_config.ini* without the *_template* ending
6. **Start *posttroll/bin/nameserver***
    - this will register the different subscribers on the network and relay the messages between different processes
    - *./nameserver*
7. **Start *trollduction/bin/trollstalker.py***
    - file watcher that sends messages of new files available for processing
    - for example: *./trollstalker.py -c /path/to/trollstalker_config.ini -C noaa_hrpt*
8. **Start Trollduction *trollduction/bin/l2processor.py***
    - *./l2processor.py -c /path/to/l2processor_config.ini -C noaa_hrpt*
    - this should print your configuration and stop to wait for new messages
9. Copy a suitable file to your data input directory
10. Check the output directory for images

CHAPTER 3

# Detailed instructions

## 3.1 Installation

You can download the trollduction source code from github,:

```
$ git clone https://github.com/mraspaud/trollduction.git
```

and then run:

```
$ cd trollduction
$ python setup.py install
```

to install. If installing system-wide, command *sudo* needs to be added before *python*, or login as user *root*. If you want to install locally on your user account, you can run instead:

```
$ python setup.py install --user
```

Trollduction is also available as a ZIP package from github, when selecting the before mentioned branch and then from the right *Download ZIP* button.

### 3.1.1 Prerequisities

If everything goes well, all the prerequisites for trollduction should be installed automatically when installing trollduction.

Here is however a list of some of the requirements for *trollduction*.

- mpop - select *pre-master* branch
- mipp - select *pre-master* branch
- pyresample - select *pre-master* branch
- posttroll - select *develop* branch
- pyorbital - select *develop* branch
- trollsift - select *master* branch
- trollduction - select *master* branch
- pytroll-schedule - select *develop* branch
- pyspectral - select *pre-master* branch
- pykdtree - select *master* branch

- python-geotiepoints - select *develop* branch
- trollimage - select *develop* branch
- pycoast - select *pre-master* branch

## 3.2 Description and operation of the different processes

Before any message-based processing, start the *posttroll* nameserver:

```
$ cd /path/to/posttroll/bin
$ ./nameserver
```

This script handles the connections between different message publishers and subscribers.

### 3.2.1 Trollstalker

Trollstalker is a script that monitors the arrival of given files in the specified directories. When such a file is detected, a pytroll message is sent on the network to notify other interested processes.

An example configuration file for trollstalker is provided in *trollduction/examples/trollstalker_config.ini_template*:

```
# This config is used in Trollstalker.

[noaa_hrpt]

# posttroll message topic that provides information on new files
# This could follow the pytroll standard:
# https://github.com/mraspaud/pytroll/wiki/Metadata
topic=/HRPT/l1b/dev/mystation

# input directory that trollstalker watches
directory=/path/to/satellite/data/

# filepattern of the input files for trollstalker
# uses the trollsift syntax:
# http://trollsift.readthedocs.org/en/latest/index.html
filepattern={path}hrpt_{platform_name}_{start_time:%Y%m%d_%H%M}_{orbit_number:05d}.l1b

# instrument names for mpop
instruments=avhrr/3,mhs,amsu-b,amsu-a,hirs/3,hirs/4

# logging config for trollstalker. Comment out to log to console instead.
stalker_log_config=/usr/local/etc/pytroll/trollstalker_logging.ini

# logging level, if stalker_log_config is not set above. Possible values are:
#  DEBUG, INFO, WARNING, ERROR, CRITICAL
loglevel=DEBUG

# inotify events that trigger trollstalker to send messages
event_names=IN_CLOSE_WRITE,IN_MOVED_TO

# port to send the posttroll messages to, optional so use "0" to take a random
# free port.
posttroll_port=0

# use an alias to convert from platform in the filename to OSCAR naming
```

```
alias_platform_name = noaa18:NOAA-18|noaa19:NOAA-19

# Keep 10 last events in history, and process only if the new event
# isn't in this history.  If option not given, or set to zero (0), all
# matching events will be processed
history=10

[hrit]
topic=/HRIT/topic/or/something/
directory=/path/to/satellite/data/
filepattern={path}H-000-{platform_name}__-{platform_name}________-_________-EPI______-{start_time:%Y%
instruments=seviri
stalker_log_config=/usr/local/etc/pytroll/trollstalker_logging.ini
loglevel=DEBUG
event_names=IN_CLOSE_WRITE,IN_MOVED_TO
posttroll_port=0
alias_platform_name = MSG2:Meteosat-9|MSG3:Meteosat-10
```

Of course, other sections can be added to the file for other files to be watched.

In order to start *trollstalker*:

```
$ cd trollduction/bin/
$ ./trollstalker.py -c ../examples/trollstalker_config.ini -C noaa_hrpt
```

Now you can test if the messaging works by copying a data file to your input directory. *Trollstalker* should send a message, and depending on the configuration, also print the message on the terminal. If there's no message, check the configuration files that the input directory and file pattern are set correctly.

### 3.2.2 l2processor

***l2processor* is the process that reads satellite data and generates composites** from it. It is triggered by messages fullfilling a given topic, reads the data file, resamples the data to given areas and generates image composites.

Before starting to configure *l2processor*, make sure that your mpop has been setup correctly (*mpop.cfg*, *areas.def*, satellite definitions). *l2processor* relies heavily on *mpop*.

To configure *l2process*, the user needs to supply at least a configuration files and a product list. The product list format is explained below.

An example configuration file for l2processor is provided in *trollduction/examples/l2processor_config.ini_template*:

```
# This config is used in l2processor

[avhrr]
# the topics in the messages to listen to.
topics=/AAPP-HRPT/1b
# the instruments we want to process
instruments=avhrr/3
# the list of products we want to generate for this type of data
product_config_file=/usr/local/etc/pytroll/product_config_hrpt.xml
# the log config file
td_log_config=/usr/local/etc/pytroll/trollduction_logging.ini
# It is sometimes possible, that the same message is received again.
# If it's not necessary to run the processing again, uncomment the
# option below, so that only first of identical consecutive messages
#  will be processed
# process_only_once=True
```

Start *l2processor* by:

```
$ cd trollduction/bin/
$ ./l2processor.py -c ../examples/l2processor_config.ini -C noaa_hrpt
```

### Product configuration file format

The product list configuration file is an xml file that contains information about the desired output of *l2processor*. An example file is provided in *trollduction/examples/product_config_hrpt.xml_template*:

```
<?xml version="1.0" encoding='utf-8'?>
<?xml-stylesheet type="text/xsl" href="prodlist2.xsl"?>

<!-- This config is used by Trollduction.-->

<product_config>
  <!-- common default values -->
  <common>
    <output_dir>/tmp</output_dir>
    <unload>False</unload>
    <!-- To remove all area coverage checks, eg. for running without
         TLE data, uncomment the following row. -->
    <!-- <check_coverage>False</check_coverage> -->
    <!-- number of processors used for parallel work -->
    <nprocs>1</nprocs>
    <!-- Use external calibration coefficients for channels 1, 2 and 3a -->
    <!-- <use_extern_calib>True</use_extern_calib> -->
  </common>


  <!-- aliases: substitutions to make in the filenames. E.g. replace
       in "platform_name" items.
  -->
  <aliases>
    <platform_name src="Metop-A" dst="metop02" />
    <platform_name src="Metop-B" dst="metop01" />
    <platform_name src="NOAA-15" dst="noaa15" />
    <platform_name src="NOAA-18" dst="noaa18" />
    <platform_name src="NOAA-19" dst="noaa19" />
    <platform_name src="EOS-Terra" dst="terra" />
    <platform_name src="EOS-Aqua" dst="aqua" />
    <platform_name src="Suomi-NPP" dst="npp" />
  </aliases>

  <!-- variables: substitution to make in the xml
       attributes. E.g. replate "output_dir" items matching with real
       path
  -->
  <variables>
    <output_dir id="path0">/san1/sir</output_dir>
    <output_dir id="path3">/san1/pps/www/latest</output_dir>
    <output_dir id="path4">/san1/pps/www/ash</output_dir>
    <overlay id="black">#000000</overlay>
    <overlay id="white">#ffffff</overlay>
  </variables>

  <!-- variables section with attribute. E.g. if the "MODE"
       environment variable is defined to "offline", the items should
```

```
      take these values instead. This takes of course precedence over
      the standart "variables" section.
-->
<variables MODE="offline">
  <output_dir id="path0">/local_disk/data/out/sir</output_dir>
  <output_dir id="path1">/local_disk/data/out/sir</output_dir>
  <output_dir id="path2">/local_disk/data/out/rgb</output_dir>
  <output_dir id="path3">/local_disk/data/out/rgb</output_dir>
  <output_dir id="path4">/local_disk/data/out/rgb</output_dir>
</variables>

<!-- Example how to group production to areas with similar coverage.
     Note that these are not implemented in the product list below!
-->
<groups>
  <group id="africa">afhorn,mali</group>
  <group id="asia">afghanistan</group>
  <group id="eport">eport</group>
  <group id="highres" unload="True" resolution="250">baws250</group>
</groups>

<product_list>
  <!-- dump to netcdf -->
  <!-- calibrated, satellite projection -->
  <dump>
    <file format="netcdf4">{time:%Y%m%d_%H%M}_{platform_name}.nc</file>
  </dump>

  <area id="eurol" name="Europe_large">
    <!-- Generate the product only if sun is above the horizon at the
         defined longitude/latitude -->
    <product id="overview" name="overview" output_dir="path0" sunzen_day_maximum="90" sunzen_lonlat
      <file output_dir="tmp">{time:%Y%m%d_%H%M}_{platform_name}_{areaname}_{productname}.png</file>
    </product>

    <!-- Generate only if the Sun is below the horizon -->
    <product id="night_overview" name="night_overview" sunzen_night_minimum="90" sunzen_lonlat="25,
      <file format="png">{time:%Y%m%d_%H%M}_{platform_name}_{areaname}_{productname}.png</file>
    </product>

    <!-- Generate also thumbnails -->
    <product id="natural" name="dnc" output_dir="path1" thumbnail_size="640x640" thumbnail_name="{p
      <file>{time:%Y%m%d_%H%M}_{platform_name}_{areaname}_{productname}.png</file>
    </product>

    <!-- add overlay using pycoast configuration "black.cfg"-->
    <product id="green_snow" name="green_snow" output_dir="path3" overlay="/usr/local/etc/pytroll/k
      <file>{time:%Y%m%d_%H%M}_{platform_name}_{areaname}_{productname}.png</file>
    </product>


  </area>

  <!-- another area -->
  <area id="euron1" name="North europe, 1km/pixel">
    <product id="red_snow" name="red_snow" sunzen_day_maximum="90" sunzen_lonlat="25, 60">
      <file format="png">{time:%Y%m%d_%H%M}_{platform_name}_{areaname}_{productname}.png</file>
    </product>
```

```
      <product id="cloudtop" name="cloudtop">
        <file format="png">{time:%Y%m%d_%H%M}_{platform_name}_{areaname}_{productname}.png</file>
      </product>

      <product id="night_fog" name="night_fog" sunzen_night_minimum="90" sunzen_lonlat="25, 60">
        <file>{time:%Y%m%d_%H%M}_{platform_name}_{areaname}_{productname}.png</file>
      </product>

    </area>
  </product_list>
</product_config>
```

The first part, <*common*>, can be used to give default values that are used, if not overridden, by all the <*product*> definitions.

The second part is <*aliases*> and contains the substitutions to perform in the file patterns (from *src* to *dst*)

The third part is <*variables*> which holds the substitutions for the tag attributes. Adding an attribute to <*variables*> checks if the corresponding environment variable is set to the given value, and uses these substitutions if it does.

The fourth part is the <*groups*> defining the area to group for processing. This means for example that the data will be loaded for the whole group (cutting at the area definition boundaries if supported). Setting th *unload* attribute to "true" provokes the unloading of the data before and after processing the group.

The next part is the <*product_list*> which contains the list of products and areas to work on.

The next layer of the product configuration is the <*area*>, which holds the following attributes:

- *name* — replaces the *{areaname}* tag in the file name template
- *id* — the name of the area/projection definition given in mpop areas.def file

The following layer is the <*product*> details to be produced in the area. The <*product*> section is given for each product. These values override the defaults given (if any) in the <*common*> section.

Required attributes within <*product*>:

- *id* — name of the function (from *mpop.image*) that produces the product
- *name* — user-defined name for the composite, this will replace the *{productname}* tag in the file name pattern
- ***overlay* — the color of the overlay to put on the image, in hex hash** (e.g. #ffffff for white) or alternatively the path to the overlay configuration file to pass to pycoast.
- *thumbnail_size* and *thumbnail_name* — the size and filename of the thumbnail to produce. The thumbnail will be written in the same directory as the image.
- *sunzen_day_maximum* — Sun zenith angle, can be used to limit the product to be generated only during sufficient lighting
- *sunzen_night_minimum* — Sun zenith angle, can be used to limit the product to be generated only during sufficient darkness
- *sunzen_lonlat* — comma-separated longitude and latitude values that can be used to define the location where Sun zenith angle values are checked. Only effective if either *sunzen_day_maximum* or *sunzen_night_minimum* is given.
- *sunzen_xy_loc* — comma-separated x- and y-pixel coordinates that can be used to define the location where Sun zenith angle values are checked. Only effective if either *sunzen_day_maximum* or *sunzen_night_minimum* is given. Faster option for *sunzen_lonlat*, but needs to be determined separately for each area.

The final layer is the <*file*> tag which holds information of the file to be saved. It can have the following attributes:

- *output_dir* — the destination directory

- *format* — the file format to use. This is optional, but if the file format cannot be easily guessed from the file extension, it's good to write it here.
- The text of this <*file*> item is the filename pattern to use.

#### Data dumps

An alternative to the <*product*> tag is the <*dump*> tag that saves the resampled data to the given filename (pattern). It can also be inserted at the previous layer to do a data dump of the unprojected data.

### 3.2.3 gatherer

Watches files or messages and gathers satellite granules in "collections", sending then the collection of files in a message for further processing.

Make sure mpop is configured. There is a template available in mpop/etc directory GDSMetop-B.cfg.template for Metop-B that can be used with gatherer.

There are several types of triggers:

```
* Observer
* PollingObserver
* Posttroll
```

Provide a gatherer configuration file.

```
[default]
regions=euron1 afghanistan afhorn

[local_viirs]
timeliness=15
duration=85.4
service=
topics=/segment/SDR/1

[ears_viirs]
pattern=/data/prod/satellit/ears/viirs/SVMC_{platform_name}_d{start_date:%Y%m%d}_t{start_time:%H%M%S
format=SDR_compact
type=HDF5
data_processing_level=1B
platform_name=Suomi-NPP
sensor=viirs
timeliness=30
duration=85.4
variant=regional

[ears_avhrr]
pattern=/data/prod/satellit/ears/avhrr/avhrr_{start_time:%Y%m%d_%H%M%S}_{platform_name}.hrp.bz2
platform_name=NOAA-19
format=HRPT
type=binary
data_processing_level=0
duration=60
sensor=avhrr/3
timeliness=15
variant=regional
```

```
[ears_metop-b]
pattern=/data/prod/satellit/ears/avhrr/AVHR_HRP_{data_processing_level:2s}_M01_{start_time:%Y%m%d%H%M
format=EPS
type=binary
platform_name=Metop-B
sensor=avhrr/3
timeliness=15
data_processing_level=0
variant=regional

[ears_metop-a]
pattern=/data/prod/satellit/ears/avhrr/AVHR_HRP_{data_processing_level:2s}_M02_{start_time:%Y%m%d%H%M
format=EPS
type=binary
platform_name=Metop-A
sensor=avhrr/3
timeliness=15
data_processing_level=0
variant=regional

[gds_metop-b]
pattern=/data/prod/satellit/metop2/AVHR_xxx_{data_processing_level:2s}_M01_{start_time:%Y%m%d%H%M%S}Z
format=EPS
type=binary
platform_name=Metop-B
sensor=avhrr/3
timeliness=100
variant=global

[gds_metop-a]
pattern=/data/prod/satellit/metop2/AVHR_xxx_{data_processing_level:2s}_M02_{start_time:%Y%m%d%H%M%S}Z
format=EPS
type=PDS
platform_name=Metop-A
sensor=avhrr/3
timeliness=100
variant=global

[regional_terra]
pattern=/data/prod/satellit/modis/lvl1/thin_MOD021KM.A{start_time:%Y%j.%H%M}.005.{proc_time:%Y%j%H%M
format=EOS_thinned
type=HDF4
data_processing_level=1B
platform_name=EOS-Terra
sensor=modis
timeliness=180
duration=300
variant=regional

[regional_aqua]
pattern=/data/prod/satellit/modis/lvl1/thin_MYD021KM.A{start_time:%Y%j.%H%M}.005.{proc_time:%Y%j%H%M
format=EOS_thinned
type=HDF4
data_processing_level=1B
platform_name=EOS-Aqua
sensor=modis
timeliness=180
duration=300
```

```
variant=regional
```

Start nameserver if it's not already running.

### 3.2.4 scisys_receiver

Receive and translates scisys ground-station message to pytroll messages.

To be written

### 3.2.5 aapp_runner

Run aapp

To be written

### 3.2.6 pps_runner

Run pps

To be written

### 3.2.7 viirs_dr_runner

Run viirs l0 -> l1 processor

To be written

### 3.2.8 modis_dr_runner

Run modis l0 -> l1 processor

To be written

CHAPTER 4

# Indices and tables

- genindex
- modindex
- search

## 4.1 Trollduction

## 4.2 Listener

Listener module for Trollduction.

**class** `trollduction.listener.`**`Listener`**(*topics=None*, *queue=None*)
PyTroll listener class for reading messages for Trollduction

**`add_to_queue`**(*msg*)
Add message to queue

**`create_subscriber`**()
Create a subscriber instance using specified addresses and message types.

**`restart`**()
Restart subscriber

**`run`**()
Run listener

**`stop`**()
Stop subscriber and delete the instance

**class** `trollduction.listener.`**`ListenerContainer`**(*topics=None*)
Container for listener instance

**`restart_listener`**(*topics*)
Restart listener after configuration update.

**`stop`**()
Stop listener.

## 4.3 XML read

XML reader for Trollduction system and product configuration files.

**class** trollduction.xml_read.**Dataset**(*data*, ***attributes*)
Dataset class.

**copy**(*copy_data=True*)
Return a copy of the data.

**class** trollduction.xml_read.**InfoObject**(***attributes*)
InfoObject class.

**get**(*key*, *default=None*)
Return the requested info.

**class** trollduction.xml_read.**ProductList**(*fname*)
Class for reading and storing product lists.

**check_groups**()
Check area groups.

**insert_vars**()
Variable replacement

**parse**()
Parse product list XML file.

trollduction.xml_read.**get_filepattern_config**(*fname=None*)
Retrieves the filepattern configuration file for trollstalker, and returns the parsed XML as a dictionary. Optional argument *fname* can be used to specify the file. If *fname* is None, the systemwide file is read.

trollduction.xml_read.**get_root**(*fname*)
Read XML file and return the root tree.

trollduction.xml_read.**parse_xml**(*tree*, *also_empty=False*)
Parse the given XML file to dictionary.

## 4.4 Helper functions

## 4.5 Custom handler

For Panu

**class** trollduction.custom_handler.**PanusTimedRotatingFileHandler**(*template*, **args*, ***kwargs*)
Like TimedRotatingFileHandler with a custom filename template.

**doRollover**()
do a rollover; If there is a backup count, then we have to get a list of matching filenames, sort them and remove the oldest ones.

**getFilesToDelete**()
Determine the files to delete when rolling over.

More specific than the earlier method, which assumed the date to be a suffix.

## t

## A



## C



## D



## G



## I



## L



## P



## R



## S



## T