pyg2p Configuration Manual

Version	1.2
Author	Domenico Nappo (domenico.nappo(at)gmail.com)
Final Release (Rev 03)	2013-10-14

Table of Contents

Introduction

The purpose of this document is to help the user to configure the **pyg2p** application and describe the content of XML configuration files and all mandatory and optional parameters.

The application is shipped with some configuration for EFAS and GLOFAS. The user might need to add parameters, conversion formulas and geopotentials for correction.

To fully configure the application, the following files must be edited (if needed):

- *parameters.xml*
- *geopotentials.xml*
- *logger-configuration.xml*

The folder ***intertables*** must exists under the ***configuration*** folder.
Geopotentials gribs, configured in *geopotentials.xml*, must be placed under the ***configuration/geopotentials*** folder.

Finally, you need to create your xml templates (path passed by -c input argument) where you configure all execution parameters.

Configuring parameters.xml file

As very first configuration, user needs to edit parameters.xml. In this file, information about parameters to extract and values conversion are being read from the application.
The form of the file must be a valid xml with the structure:

```
<Parameters>
      <Parameter...>
            <Conversion …/>
      </Parameter>
      <Parameter …/>
<Parameters>
```

A Parameter element has the following attributes (mandatory attributes are in **bold**):

Attribute/Tag	Details
shortName	The short name of the parameter, as it appears in the grib file. The application uses this for selecting messages.
description	
unit	
Conversion	Xml tag node [0..*] describing conversion and cut off of negative values.

Here below the attributes to use for the Conversion element:

Attribute/Tag	Details
id	The unique name to identify the conversion within the set of conversions defined for a same parameter. It's being used in execution xml file configuration as *applyConversion* attribute.
unit	Target unit
function	Function for conversion. **Must be in the form *x=f(x). E.g.: x=x-273.15***
cutOffNegative	Boolean.Negative values will be set to 0 if True.

Example:

```
<Parameters>
      <Parameter shortName="fis" description="Geopotential" unit="m^2/s^2" />
      <Parameter shortName="sro" description="Surface Runoff" unit="m">
                  <Conversion id="m2mm" unit="mm" function="x=x*1000"
cutOffNegative="True" />
      </Parameter>
       <Parameter  shortName="alhfl_s" description="latent heat flux" unit="W/
(m^2)">
            <Conversion id="tommd" unit="mm/d" function="x=x*(-0.0353)"
cutOffNegative="True" />
            <Conversion id="tommh" unit="mm/h" function="x=x*(-0.00147)"
cutOffNegative="True" />
      </Parameter>
      <Parameter shortName="2D" description="2 meters Dewpoint" unit="K">
            <Conversion id="k2c" unit="C" function="x=x-273.15"/>
      </Parameter>
      <Parameter shortName="2t" description="2 meters Temperature" unit="K">
            <Conversion id="k2c" unit="C" function="x=x-273.15"
cutOffNegative="False" />
      </Parameter>
</Parameters>
```

Configuring loggers

The file logger-configuration.xml in the folder *configuration* will serve to setup loggers levels, logfiles full path and format. It's important to not change the formatting string, unless aware of the

consequences. In no case, the name attribute of the logger must be changed.
File logging can be disabled setting the optional attribute *enabled* within the *Loggers* element to one of the values false, False, No, NO, no (default is True):

```
<Loggers enabled="false">
```

Accepted levels are: “DEBUG”, “INFO”, “WARN”, ”WARNING”, “ERROR”.
Use the placeholders *%(asctime)s*, *%(message)s*, and *%(levelname)s* to personalise the output format with the *formatting* attribute.
Keep attention to not accidentally change *name* attribute.

Some examples:

```
<Logger file="GRIBReader.log" level="DEBUG" name="GRIBReader"
            formatting="[%(asctime)s] : %(levelname)s %(message)s" />
```

Console logging level is INFO by default. This value can be overwritten by command line input argument -l (or --loggingLevel=)
Default logs' output dir is ./logs/. It can be overwritten with -d (--outLogDir=):

```
/pyg2p_svn/pyg2p.py -c /pyg2p_svn/execution_templates/eue_t24.xml -i
/dataset/test_2013330702/EpsN320-2013063000.grb -o ./eue -m 10 -l ERROR -d
/dataset/logs_20130705
```

Configuring geopotentials.xml file

The file geopotentials.xml under configuration folder, is used to get the right geopotential filename to correction routines. The application read the xml file searching for an id, which has got the form:

longFirstDegrees$longLastDegrees$NiNjnumberOfValues$gridType

E.g.: -35$45$161$91$14651$regular_ll

To get the id from a grib file use:

```
grib_get -p
longitudeOfFirstGridPointInDegrees,longitudeOfLastGridPointInDegrees,Ni,Nj,numberOf
Values,gridType /path/to/grib.grb
```

and get the filename of the geopotential grib to use.

The application will read the right geopotential file reading those geodetic keys from the input grib file. So it's important that the geopotential you want to use for correction must have the same grid of the input grib message to correct.

A geopotentials.xml file is quite simple. Grib geopotentials files must be under configuration/geopotentials directory.

```
<geopotentials>
```

```
    <!-- id has the form:
        longFirstDegrees$longLastDegrees$Ni$Nj$numberOfValues$gridType
    -->
 <geopotential id="-35$45$161$91$14651$regular_ll"
name="europe_0.5_degree_T799.grb"/>
    <geopotential id="-35$45$81$46$3726$regular_ll"
name="europe_1_degree_T399.grb"/>
…
…
<geopotential id="-18$23.5$665$657$436905$rotated_ll"
name="dwd_grib1_ispra_LME_2013051300"/>
</geopotentials>
```

Configuring execution template files

The application will run via CLI using the command:

./pyg2p.py -c <path/to/execution_file.xml> [-o /output/dir] -i file.grib [-I file2ndres.grib] [-s <tstart>] [-e <tend>] [-m <epsMember number>] [-T <0|1200> [-l <level of console logger>] [-d /log/dir] [-t test.xml]

This section is dedicated to writing *execution xml* files.

For some examples of commands files, see into the folders execution_file_examples/ and execution_templates/.

The typical content of an execution.xml file is reported below:

```
<?xml version="1.0" encoding="utf-8"?>

<Execution name = "test3" id="103"  >
 <Parameter shortName="td_2m" tstart="12" tend="18"
            applyConversion="k2c"
            gem="(z/9.81)*0.0065"
            correctionFormula="p+gem-dem*0.0065"
            demMap="/maps/dem.map"

 />
 <Aggregation type="average" step="24" />
 <OutMaps namePrefix="td2" fmap="001" cloneMap="/maps/dem.map" ext=”1”>
     <Interpolation mode="grib_invdist"
                    latMap="/maps/lat.map"
                    lonMap="/maps/long.map"
     />
  </OutMaps>
</Execution>
```

Execution tag

Attribute/Tag	Details
`name`	The short name of the parameter, as it is in the grib file. The application use this for selecting messages.
`id`	
`Parameter`	XML Tag. See relative table
`Aggregation`	XML Tag. See relative table
`OutMaps`	XML Tag. See relative table

Parameter tag

Attribute/Tag	Details
`shortName`	The short name of the parameter, as it is in the grib file. Must be configured in the parameters.xml file, otherwise the application exits with an error. Main grib selector.
`tstart`	Optional grib selectors tstart, tend, and dataTime can also be issued via command line arguments (-s, -e, -T), which overwrite the ones in the execution xml file. *Note that the perturbationNumber selector is given by -m command line input argument.*
`tend`	
`dataTime`	
`level`	
`applyConversion`	The conversion id to apply, as in the parameters.xml file for the parameter to select. The combination parameter/conversion must be properly configured in parmaters.xml file, otherwise the application exits with an error.
`correctionFormula`[1]	Formula to use for parameter correction with p, gem, dem variables, representing parameter value, converted geopotential to gem, and DEM map value. E.g.: p+gem*0.0065-dem*0.0065
`demMap`	The dem map used for correction.
`gem`	Formula for geopotential conversion for correction.

1 When configuring correction, all xml attributes *gem*, *correctionFormula*, and *demMap* must be present.

OutMaps tag

Attribute/Tag	Details
cloneMap	The clone map with area (must have a REAL cell type and missing values for points outside area of interest. A dem map works fine. A typical area boolean map will not).
unitTime	Time unit in hours of output maps. Tipical value is 24 (daily maps).
namePrefix	Prefix for maps. Default is parameter *shortName*.
fmap	First map number. Default 1.
Interpolation	XML tag. See relative table.
ext	Extension mode. It's the integer number defining the step numbers to skip when writing maps. Same as old grib2pcraster. Default 1. Refers to User Manual for further information.

Aggregation tag

Attribute/Tag	Details
step	Step of aggregation in hours.
type	Type of aggregation (it was Manipulation in grib2pcraster). It can be *average* or *cumulation*.

Interpolation tag

Attribute/Tag	Details
mode	Interpolation mode. Possible values are: “griddata”, “nearest”, “invdist”, “grib_nearest”, “grib_invdist”
latMap	PCRaster map of target latitudes.
lonMap	PCRaster map of target longitudes.
intertableDir	Alternative home folder for interpolation lookup tables, where pyg2p will load/save intertables. Folder must be existing.
method	Used only for mode=”griddata”. It can be “linear”,

	“nearest”, “cubic”. Currently, during tests, only “nearest” method produce correct maps. “linear” and “cubic” methods could produce maps which cannot be opened by PCRaster Aguila tool.
`p`	Used only for mode=”invdist”. It can be “1” or “2”. Default will be 1.
`leafsize`	Used only for mode=”invdist”. Default 10.
`eps`	Used only for mode=”invdist”. Default 0.1.

Configuring the application for test functionality

Since version 1.1, the application has an additional input argument *-t <path/to/file.xml>*which will perform the following:

- execute a set of predefined grib2pcraster commands
- execute a set of the corresponding pyg2p commands with the same options of grib2pcraster
- produce diff PCRaster maps for manual comparison between grib2pcraster and the new pyg2p version.
- Output in console the aguila commands for opening maps

The application comes with preconfigured test.xml and commands.txt files. You must edit the paths and the commands XML templates used. In the following paragraphs, there are simple instructions to write your own tests.

Writing an XML configuration for tests

The xml file used by the test functionality is quite easy. It just has to contain paths to:

- grib2pcraster executable
- pcrcalc executable
- text file containing list of commands in pairs

Just use the following template:

```
<TestConfiguration
commands="/ECMWF_grib2pcraster/pyg2p_workingcopy/configuration/tests/commands.txt">
    <g2p exec="/ECMWF_grib2pcraster/grib2pcraster/grib2pcraster" />
    <PcRasterDiff exec="/devapps/PCRaster-3.0/bin/pcrcalc"/>
</TestConfiguration>
```

Writing the commands.txt file

Writing the commands.txt file is also very easy. You need to write pairs (or triples) of arguments lists,

one (or two) for grib2pcraster and one for pyg2p, **each pair/triple identifying a single test case**. For each test case, pcrcalc diff maps will be produced.

Each pair/triple of arguments lists looks like the following:

gN@ <list of arguments for grib2pcraster>
gN@ <list of arguments for the second optional grib2pcraster execution>
pN@ <list of arguments for pyg2p>

where N is an integer id. You exactly have to use the prefix '**g**' and '**p**' to identify the command for grib2pcraster and pyg2p respectively.
The second optional test execution of grib2pcraster is needed for comparison tests in spatial multiresolution files (i.e. glofas executions).

An example:

```
g1@ -p 228.128 -n gR24 -t 24 --fmap=1 -i /dataset/EpsN320-2013063000.grb -o
/dataset/eueR24 -m 10
p1@ -l ERROR -c /ECMWF_grib2pcraster/pyg2p/execution_templates/eue_r24.xml -i
/dataset/EpsN320-2013063000.grb -o /dataset/eueR24 -m 10

g2@ -p 182.128 -n gE24 -t 24 --fmap=1 -m 10 -i /dataset/EpsN320-2013063000.grb
-o /dataset/eueE24
p2@ -l ERROR -c /ECMWF_grib2pcraster/pyg2p/execution_templates/eue_e24.xml -i
/dataset/EpsN320-2013063000.grb -o /dataset/eueE24 -m 10

#glofas commands. grib2pcraster needs two executions for each pyg2p command
g3@ -i /dataset/20130325_en0to10.grib -o ./multiresSRO -p 8.128 -m 50 -n gsro
--date=20130325 -s 0 -e 240 --unit=mm -t 24
--interlookup=/grib2pcraster/interplookup/intertable_2011_medium.txt
g3@ -i /dataset/multires/20130325_en11to15.grib -o ./multiresSRO -p 8.128 -m 50 -n
gsro --date=20130325 -s 264 --fmap=11 -e 360 --unit=mm -t 24
--interlookup=/grib2pcraster/interplookup/intertable_2011_low.txt
p3@ -c /pyg2p_workingcopy/execution_templates_devel/glofas_sro.xml -i
/dataset/20130325_en0to10.grib -I /dataset/20130325_en11to15.grib -o ./multiresSRO
-m 50

#only pyg2p tests (these are not reproducible with grib2pcraster 1.07)
p4@ -i /dataset/UKMO/UKMO_20121218.grib -o ./UKMO/ -c
/home/dominik/ECMWF_grib2pcraster/pyg2p_workingcopy/execution_templates_devel/UKMO_
t24_LA.xml -m 0
p4@ -i /dataset/ungrib_20130804/EUE_2013073100_cv.grb -o ./UKMO/ -c
./execution_templates_devel/EUE_RainAnim_CV.xml -m 0
```

Important notes:

You have to use the prefix **g** in the filename of maps produced by **grib2pcraster** and the prefix **p** for those ones produced by **pyg2p** (this must be configured in the commands xml template **e.g.:** eue_r24.xml).
Output directory must be the same within each test case (*-o /dataset/eueE24* for boths).