The EUMETSAT Network of Satellite Application Facilities

NWC SAF
Support to Nowcasting and Very Short Range Forecasting

User Manual for the NWC/PPS application: Software Part, 1.Installation

NWC/CDOP2/PPS/SMHI/SW/UM/1, Issue 1, Rev. 0

15 September 2014

Applicable to SAFNWC/PPS version 2014

Applicable to the following PGE:s:

PGE	Acronym	Product ID	Product name	Version number
PGE01	CM	NWC-062	Cloud Mask	4.0
PGE02	CT	NWC-065	Cloud Type	2.0
PGE03	CTTH	NWC-068	Cloud Top Temperature and Height	4.0
PGE04	PC	NWC-073	Precipitating Clouds	1.6
PGE05	CPP	NWC-071	Cloud Physical Properties	1.1

Prepared by Swedish Meteorological and Hydrological Institute (SMHI)

REPORT SIGNATURE TABLE

Function	Name	Signature	Date
Prepared by	SMHI		15 September 2014
Reviewed by	SAFNWC Project Team EUMETSAT		9 September 2014
Authorised by	Anke Thoss, SMHI *SAFNWC PPS Manager*		15 September 2014

DOCUMENT CHANGE RECORD

Version	Date	Pages	Changes
1.0d	3 July 2014	62	This document (NWC/CDOP2/PPS/SMHI/SW/UM/1) together with NWC/CDOP2/PPS/SMHI/SW/UM/2 are replacing CDOP-document: SAF/NWC/CDOP/SMHI-PPS/SW/SUM/1 First version for SAFNWC/PPS v2014 Implemented RIDs from PCR-v2014: -LSc2 (formal issues) -Action 8 (note on new output format) General changes for v2014: -Added description of binary installation. -Removing description of Task Manager. -Updating configurations.
1.0	15 September 2014	60	Implemented RIDs from DRR-v2014: -TH: typos and grammar General changes: -Updating packages in binaries for CentOS (closing part of TBD-02). -Removed py_packages. -Added some more information about binaries. -Added a missing section for netCDF4, in Third party software. -Minor changes due to less dependence of AAPP (absolute azimuth angles and AAPP_PREFIX). -Minor changes due to a new call to python scripts (GlobalMetop; added argparse in list of Third party software) -Already existing TBD:s are now properly listed.

Table of Contents

List of Tables and Figures

1. INTRODUCTION

The EUMETSAT "Satellite Application Facilities" (SAF) are dedicated centres of excellence for processing satellite data, and form an integral part of the distributed EUMETSAT Application Ground Segment (http://www.eumetsat.int). This documentation is provided by the SAF on Support to Nowcasting and Very Short Range Forecasting, SAFNWC. The main objective of SAFNWC is to provide, further develop and maintain software packages to be used for Nowcasting applications of operational meteorological satellite data by National Meteorological Services. More information can be found at the SAFNWC webpage, http://www.nwcsaf.org . This document is applicable to the SAFNWC processing package for polar orbiting meteorological satellites, SAFNWC/PPS, developed and maintained by SMHI (http://nwcsaf.smhi.se).

1.1 NEW FILE FORMAT AND NAMING!

Before you start installing the PPS software, it is important to acknowledge and be prepared that the v2014 of PPS will produce output, in terms of file names and file format, that has significantly changed compared to previous versions of PPS.

So compared to PPS v2012 (and any previous version of PPS), the file names will differ, as well as the names of the dataset, the class definitions (CMa, CT and CPP-phase), the processing flags and the attributes. However, the physical parameters or products will be the same, and all dataset that were present in v2012 (except processing flags) will also be present in the files of v2014. For details on the new data format see Data Output Format [RD.11].

1.2 PURPOSE

This document presents the Software User Manual for the SAFNWC/PPS application. The main objective of the SAFNWC/PPS software is the generation of 5 meteorological products from polar orbiting satellites for the support to Nowcasting and Very Short Range Forecasting.

The different Product Generation Elements (PGEs) have been designed in such a way as to allow individual execution as stand alone applications, though depending on input from each other.

This Software User Manual is supposed to contain all the information necessary to build and install the whole SAFNWC/PPS application.

1.3 SCOPE

1.4 DEFINITIONS AND ACRONYMS

Acronym	Explanation	Acronym	Explanation
AAPP	AVHRR and ATOVS Processing Package	**AEMET**	Agencia Estatal de Meteorología (Spain)
ACPG	AVHRR/AMSU Cloud Product Generation software (A major part of the SAFNWC/PPS s.w., including the PGE:s.)	**AHAMAP**	AMSU-HIRS-AVHRR Mapping Library (A part of the SAFNWC/PPS s.w.)
		AMSU	Advance Microwave Sounding Unit

Acronym	Explanation
ANA	Automatic Navigation Adjustment
API	Applications Programming Interface
API	Application Programming Interface
ATOVS	Advanced TOVS
AVHRR	Advanced Very High Resolution Radiometer
CDOP	Continuous Development and Operational Phase
CM	Cloud Mask (also PGE01)
CMS	Centre Météorologique Spatial (a part of Météo-France)
CPP	Cloud Physical Properties
CT	Cloud Type (also PGE02)
CTTH	Cloud Top Temperature, Height and Pressure (also PGE03)
ECMWF	European Centre for Medium-range Weather Forecasts
EPS	EUMETSAT Polar System
EUMETCast	EUMETSAT's Broadcast System for Environmental Data
EUMETSAT	European Organisation for the Exploitation of Meteorological Satellites
GUI	Graphic User Interface
HDF5	Hierarchical Data format version 5
HIRLAM	High Resolution Area Model
HLHDF	High Level interface for HDF5
HRPT	High Resolution Picture Transmission
IASI	Infrared Atmospheric Sounding Interferometer
INM	Instituto Nacional de Meteorología (Spain) (old name of AEMET)
I/O (or IO)	Input/Output
IR	Infrared
LST	Land Surface Temperature
MHS	Microwave Humidity Sounding Unit
MODIS	Moderate Resolution Imaging Spectrometer
MSMS	Matching Satellite Model and Synop
MW	Microwave
NA	Not Applicable
NCSA	National Center for Supercomputing Applications (US)
NOAA	National Oceanic and Atmospheric Administration
NWP	Numerical Weather Prediction
OS	Operating System
OSISAF	Ocean and Sea Ice SAF
PC	Precipitating Cloud (also PGE04)
PCPN	Precipitation
PGE	Process Generating Element
PPS	Polar Platform System
RGB	Red Green Blue
RTM	Radiative Transfer Model
SAF	Satellite Application Facility
SAFNWC	Satellite Application Facility for support to NoWcasting
SC-EM	Screen and Email (an option for logs)
SMHI	Swedish Meteorological and Hydrological Institute
SST	Sea Surface Temperature
SUM	Software User Manual
SW	SoftWare
TBUS	TIROS Bulletin United States (contains predicted orbital elements)
TM	PPS Task Manager
TOA	Top Of Atmosphere
USGS	U.S. Geological Survey
UTC	Universal Time Co-ordinated
VIS	Visible
WMO	World Meteorological Organisation

See [RD.1] for a complete list of acronyms for the SAFNWC project.

1.5 REFERENCES

1.5.1 Applicable documents

The following documents, of the exact issue shown, form part of this document to the extent specified herein. Applicable documents are those referenced in the Contract or approved by the Approval Authority. They are referenced in this document in the form [AD.X]

For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the current edition of the document referred applies.

Current documentation can be found at SAFNWC Helpdesk web: http://www.nwcsaf.org

Ref	Title	Code	Vers	Date
[AD.1.]	Proposal for the Second Continuous Development and Operations Phase (CDOP) March 2012 – February 2017	NWC/CDOP2/MGT/AEMET/PRO	1.0	15/03/11
[AD.2.]	NWCSAF Project Plan	NWC/CDOP2/SAF/AEMET/MGT/PP	1.3	29/11/13
[AD.3.]	Configuration Management Plan for the NWC SAF	NWC/CDOP2/SAF/AEMET/MGT/CMP	1.2	29/11/13
[AD.4.]	NWCSAF Product Requirements Document	NWC/CDOP2/SAF/AEMET/MGT/PRD	1.5	05/06/14
[AD.5.]	System and Components Requirements Document for the NWC/PPS	NWC/CDOP2/PPS/SMHI/SW/SCRD	1.0	15/09/14

Table 1: List of Applicable Documents

1.5.2 Reference documents

The reference documents contain useful information related to the subject of the project. These reference documents complement the applicable ones, and can be looked up to enhance the information included in this document if it is desired. They are referenced in this document in the form [RD.X]

For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the current edition of the document referred applies

Current documentation can be found at SAFNWC Helpdesk web: http://www.nwcsaf.org

Ref	Title	Code	Vers	Date
[RD.1]	The Nowcasting SAF Glossary	NWC/CDOP2/SAF/AEMET/MGT/GLO	2.0	18/02/14
[RD.2]	User Manual for the NWC/PPS application: Software Part, 2. Operation	NWC/CDOP2/PPS/SMHI/SW/UM/2	1.0	15/09/14
[RD.3]	System Version Document for Third Part Software for the NWC/PPS	NWC/CDOP2/PPS/SMHI/SW/SCVD/ThirdParty	1.0	15/09/14
[RD.4]	Interface Control Document for Internal and External Interfaces of the NWC/PPS	NWC/CDOP2/PPS/SMHI/SW/ICD/1	1.0	15/09/14
[RD.5]	Algorithm Theoretical Basis Document for Cloud Mask of the NWC/PPS	NWC/CDOP2/PPS/SMHI/SCI/ATBD/1	1.0	15/09/14
[RD.6]	Algorithm Theoretical Basis Document for Cloud Type of the NWC/PPS	NWC/CDOP2/PPS/SMHI/SCI/ATBD/2	1.0	15/09/14
[RD.7]	Algorithm Theoretical Basis Document for Cloud Top Temperature Pressure and Height of the NWC/PPS	NWC/CDOP2/PPS/SMHI/SCI/ATBD/3	1.0	15/09/14
[RD.8]	Algorithm Theoretical Basis Document for Precipitating Clouds of the NWC/PPS	NWC/CDOP2/PPS/SMHI/SCI/ATBD/4	1.0	15/09/14
[RD.9]	Algorithm Theoretical Basis Document for Cloud Physical Properties of the NWC/PPS	NWC/CDOP2/PPS/SMHI/SCI/ATBD/5	1.0	15/09/14
[RD.10]	User Manual for the NWC/PPS Application: Science Part	NWC/CDOP2/PPS/SMHI/SCI/UM/1	1.0	15/09/14
[RD.11]	Data Output Format for the NWC/PPS	NWC/CDOP2/PPS/SMHI/SW/DOF	1.1	15/09/14
[RD.14]	The AMSU-HIRS_AVHRR Mapping Library (AHAMAP) User Manual		0.63	May 2002
[RD.15]	Improved navigation of Advanced Very High Resolution Radiometer data at high latitudes	SMHI-BPF/TEC/AVHRR/1	1.0	21/12/04
[RD.16]	Brunel, P. and Marsouin, A., 2000. Operational AVHRR navigation results. *Int. J. of Remote Sensing*, **21**, 951-972.			2000

Table 2: List of Referenced Documents

1.6 DOCUMENT OVERVIEW

This document contains the information required to install the SAFNWC/PPS application. The document is divided in the following sections:

- Section 1: Introduction and list of reference and applicable documents
- Section 2: Provides a system overview.
- Section 3: Describes this manual and how to read it
- Section 4: Provides an overview of tested platforms and OS's
- Section 5: Describes how to install the PPS-package using binaries.
- Section 6: Describes how to install the PPS-package using source code.
- Section 7: Provides a software overview.
- Section 8: Describes configuration of the environment
- Section 9: Describes how to solve some common problems related to installation.
- Appendix A: a list on TBD:s and TBC:s
- Appendix B: Describes required compilers and third party software (pre-requisite for the PPS-package)

2. SOFTWARE INTRODUCTION

2.1 SYSTEM OVERVIEW

SAFNWC/PPS implements the functionalities required to generate the SAF products for Nowcasting and very Short Range Forecasting from polar orbiting satellite data. The core of the SAFNWC/PPS is constituted by stand-alone applications responsible to generate the different meteorological products.

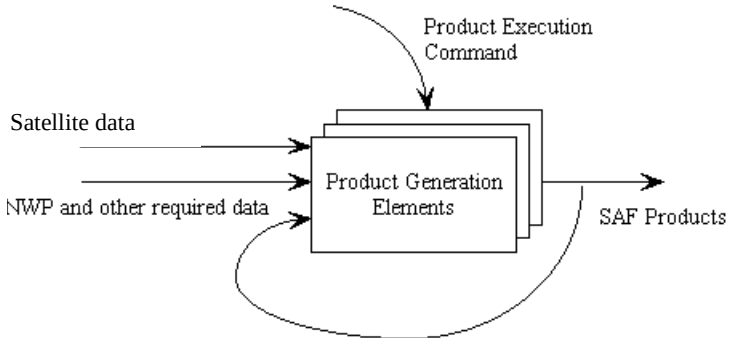

Figure 1: Simplified SAFNWC/PPS design

The standard use of SAFNWC/PPS will be the timely generation of near real-time products. The frequency for this generation cycle is dictated by the frequency of the local reception of EPS Metop, NOAA and Suomi NPP satellite overpasses[1]. In such an operational mode it is necessary to be aided by an automatic application in charge of executing correctly the sequence of tasks needed to process the satellite data. It can be done by a crontab-solution type scheduler, or the task can be performed by a scheduler mechanism provided by the user, calling the PPS modules in the appropriate order or calling a PPS master script (see [RD.2]).

SAFNWC/PPS has been designed in such way to allow a full configuration. That means that the implementation of the application includes minimum information about processing of the PGEs and secondary actions. All this data is submitted in several configuration files, and therefore it can be configured according to the user's preferences.

2.2 LICENSE AND CONDITIONS OF USE

The software accompanying this Users Manual is provided under license. Rights to use, copy, or modify, this software follows EUMETSAT policy of the SAFNWC/PPS software, and is specified in the dedicated license agreement.

A Help Desk facility is available for the registered user. The exact coordinates of this Help Desk website is http://www.nwcsaf.org .

[1] Later also JPSS-1 and 2.

3. DESCRIPTION OF THE SOFTWARE USER MANUAL

3.1 HOW TO USE THIS MANUAL

The User Manual (UM) describes the tasks to be performed in order to use the SAFNWC/PPS software package properly. This document is part one (UM/1) of the overall PPS Software User Manual and describes the installation and building procedure of the software, and provides a list of environment variables needed.

The second part of the User Manual (UM/2 [RD.2]) provides instructions for operations, including configuration and need of input data.

3.2 WHO SHOULD READ THIS MANUAL

For the installation and operation of the PPS-software package, and the use of its end products we operate with three different kinds, or groups, of users:

- An administrative user – *sys-admin*
- A scientific user – *science-admin*
- An end user – *end-user*

Typically the sys-admin would be the person building and installing the software, and responsible for setting up and running the software in an operational real time environment. The science-admin would be a researcher/scientist operating the software, or part of it, in an off-line mode on archived data in a development environment. And finally, the end-user is the forecaster using the four PPS products.

This software user manual is intended first of all for the person in charge of building and installing the PPS software package, thus the sys-admin.

For the science-admin, the other part of the Software User Manual, 2.Operation [RD.2] is more relevant, describing how to change configuration parameters, and of course how to invoke the different components of the package.

This document is not intended for the end-user. For the end-user, we refer to the Scientific User Manual [RD.10], the Algorithm Theoretical Basis Documents ([RD.5], [RD.6], [RD.7], [RD.8] and [RD.9]). The ATBDs will of course also be relevant for the science-admin.

3.3 CONVENTIONS USED IN THIS MANUAL

The following typographical conventions are used in this document:

Italic

Are used for words or phrases in a computer code expression; that shall be substituted with an appropriate real value for the expression to be valid. It is also used for terms the first time they are appear or simply to emphasise a word or a term.

`Constant width` or

```
Code formatting
```

Are used for real filenames, directory names, and computational commands and output on the screen (or in a log file).

4. PLATFORMS AND OPERATING SYSTEMS

The PPS-software is written almost exclusively in Python (2.x compatible) and ANSI C. There is only one interface written in Fortran 90 for the execution of the RTTOV radiative transfer model needed for the CTTH product (PGE03). As such the software should compile on most Linux/Unix systems provided there is a compiler and linker for C and Fortran 90.

Historically PPS (version 2.0 and later) has been built on SunOS, IBM AIX and various Intel PC Linux'es (Mandrake, Radhat, Debian, Ubuntu, both 32 bit and 64 bit architectures). (See System Version Document ([RD.3]) for details on versions.)

PPS v2014 (and v2012) is running at SMHI on RedHat Enterprise Linux version 6.x (64 bit architecture). With version 2014 PPS binary distributions ready for a few Linux systems (RedHat RHEL 6, CentOS 6, Suse SLES 11, OpenSuse (version TBD-02) and Ubuntu (version TBD-02)) are provided. Both binary packages (rpm's or debian) and source packages are available. See 5.3 for further details.

4.1 REQUIRED RESOURCES

The system requirements for running the PPS software depend somewhat on the setup and local reception. Running PPS on locally received Suomi NPP VIIRS data will probably require the most in memory and computational power, once the scene has been acquired. But running only on local data will mean the system has some time to finish the scene before the next one arrives. On the other hand, if running PPS on EARS data or the Global Metop GDS data stream will require a constant feed of data granules over the day. In such a setup PPS needs to at least be able to process all products on one x-minute granule in x minutes. Thus for EARS-AVHRR 1 minute of AVHRR data (380 AVHRR scans) needs to be processed in less than 1 minute. For Metop GDS 3 minutes of Metop EPS level1a data needs to be processed in less than 3 minutes. With todays CPU power, it is our experience that this is only achievable using parallel processing and utilising several CPUs.

PPS scales well if the data are in granules, as it is easy to setup PPS to run on several granules in parallel. However, on one chunk of data (an entire local pass or a granule) PPS scales only slightly, as there is no internal sub-granule segmentation. The script ppsRunAllParallel.py runs all PGEs in as parallel processing as possible, see [RD.2] for details.

Below we provide a rough guide for the minimum requirements for the running of PPS operationally on direct readout data (AVHRR and VIIRS):

- 8 Gb RAM (16 recommended)
- 100 Gb free hard disk (0.5 Tb recommended)
- Local access to NWP (ECMWF, or other global or regional NWP model) data
- Access to locally received AAPP AVHRR level 1b or VIIRS SDR (CSPP or IPOPP processed) and MHS/AMSU-A/B level 1c (AAPP processed)
- Optionally, access to AVHRR or VIIRS data via EUMETCast (e.g. EARS-AVHRR, EARS-VIIRS or Metop GDS).

5. BINARY INSTALLATION

From version 2014 the NWCSAF/PPS software and the most essential 3'rd party SW packages will be available also in binary form to ease installation on the user side. PPS-v2014 is available both for RPM and Debian based Linux systems. We have focused on just a few of the most commonly used platforms, but we intend to add platforms depending on the user-requests. Through the availability of ready made source packages it will also be easier for the user to re-build PPS for a specific platform, should binary packages not be available yet for the desired system.

To start with, PPS is available for CentOS-6.4, RedHat RHEL6.5, SUSE SLES 11, Open Suse (version TBD-02) and Ubuntu (version 14.04 and TBD-02) (TBD-02).

Section 5.1 describes the procedure to download and install the binary distribution and section 5.2 describes how to verify the installation. We have used the CentOS built as the example here, but the overall procedure is the same independent of the actual distribution, only the name of the install scripts differ, and of course the binary packages.

In section 5.3 we describe the binary distributions supported currently, and what may be important to know about them before installation. Please shortly consult the section corresponding to your distribution before continuing below.

Before you start: Be prepared that the v2014 of PPS will give output in a different format than PPS v2012. See 1.1 for a short note on that.

5.1 SET UP, DOWNLOAD AND INSTALL

First create the following directories locally e.g. under the user's (the user that is going to run pps) home directory:

- `ppswork`
- `data`

Make sure you have a minimum of 15 Gb (20 Gb recommended) free disk space before continuing!

NB! Also make sure that the user which is going to install the PPS software is included in the sudoers list!

Then download the tar file with all the RPM packages and unpack them into the directory RPMS (exact procedure for the debian packages for Ubuntu TBD-05). Also download the install and verification scripts and place them in the *ppswork* directory:

- pps_install_scripts.tgz

The scripts relevant for CentOS are these:

- centos64_yum_install_pps.sh

- centos64_yum_remove_pps.sh
- pps_check_output.py
- pps_test.sh
- pps_test_metopgds.sh

Also download the ancillary data for the CPP algorithm:

- cpp_ancdata_v2014.tgz

These data are not part of any rpm package due to the large amount of data (3Gb).

Now you shall copy the pps run scripts (pps_test.sh and pps_test_metopgds.sh) and the test data check script pps_check_output.py to the ppswork dir.

In order to eventually check your local implementation you can check the results of PPS run locally (on both the global and the direct readout Metop-b data) against a reference dataset. Thus please download the reference data as well and unpack them in the data directory:

```
%> cd ${HOME}/data
%> tar xzf ../pps_refdata.tgz
```

Before installing clean up your environment by running the remove script above.

Then start the installation by running the install script from the directory ${HOME}/RPMS (if you are on CentOS):

```
%> centos64_yum_install_pps.sh
```

Replace the centos64 with suse_sles11 or something else adequate for your platform.

5.2 TEST RUN AND VERIFICATION OF INSTALLATION

5.2.1 Prepare the test run

The rpm packages nwcsaf_acpg-data and nwcsaf_acpg-testdata contain all ancillary data and testdata to run PPS, except the two very large ancillary data files for the CPP products (cpp_lut.5chan.16bit.v<date>.h5 and cpp_anc_clim.v<date>.h5, where <date> depends on the version -use the ancillary data delivered with the version you are using.).

The rpm packages creates a data structure under /usr/share/pps. But as this data structure will belong to root and only have read access for the user running PPS you should make a copy of the data to a local directory structure. Therefore go to the data directory created earlier and get the data:

```
%> cd $HOME/data
%> rsync -a /usr/share/pps/ .
```

(You can now uninstall the data and test data packages if you want to save disk space).

Now go to where the CPP ancillary data files are expected and unpack the tar file there:

```
%> cd $HOME/data/import/CPP_data/source
%> tar xzf cpp_ancdata_v2014.tgz
```

This will take a little while (~3 Gb of data). Be sure to have enough disk space available.

Now go to the `ppswork` directory. Copy the file with pps environment parameters and adapt it to your local data structure (see below):

```
%> cp /opt/acpg/cfg/.profile_pps
```

Edit the file, change the DATA_DIR and PFS_LVL1_DIR variables and the PYTHONPATH settings, pointing to where you have the data and the absolute path to your local PPS settings. For example:

```
%> DATA_DIR=${HOME}/data ; export DATA_DIR
%> PFS_LVL1_DIR=${HOME}/data/import/PPS_data/source ; export PFS_LVL1_DIR
```

5.2.2 Go for it!

Now run the full suite of PPS PGE's on the Metop-B test scene:

```
%> ${HOME}/ppswork/pps_test.sh
```

And run the cloud mask and type on the Global Metop test granule (EPS format):

```
%> ${HOME}/ppswork/pps_test_metopgds.sh
```

5.2.3 Verification

Data should be created under

`${HOME}/data/export`

You can verify the results by comparing to the reference results. E.g.:

```
%> cd {HOME}/ppswork
%> python ./pps_check_output.py ${HOME}/data/export {HOME}/data/ref_results
```

The scripts checks for the equality of the images of each PGE output against a reference dataset. If everything is ok, the script will for each image write a line ending with OK plus a number or the text string `inf`. To check the equality of the images the script uses the `ImageMagick` command `compare` and the image metric `psnr`. If the `psnr` score get's too low the script will terminate raising a `ValueError` and informing you about the result. If the images are exactly the same, the `compare` script will output the result `inf`. To allow for numerical deviations depending on differences in the OS and hardware between the reference platform (RHEL-6.4 at SMHI) where the reference data has been produced, and the local OS and hardware, the verification script allows for a certain maximum deviation between the test and the reference images.

5.3 DISTRIBUTIONS

5.3.1 CentOS

NB! The rpm packages to be downloaded are all gathered in one big tar file!

The following rpm packages of PPS are available:

- nwcsaf_acpg-data-v2014+20141007-1.x86_64.rpm
- nwcsaf_acpg-testdata-v2014+20141007-1.x86_64.rpm
- nwcsaf_acpg-v2014+20141007-1.x86_64.rpm
- nwcsaf_cpp-v2014+20141007-1.x86_64.rpm

- nwcsaf_ahamap-v2014+20141007-1.x86_64.rpm
- pps_nwp-0.4.10-1.el6.noarch.rpm

In addition these 3rd party rpm packages specifically needed by PPS are included in the package tar file:

From the CentOS repository or ready built elsewhere	Built by the PPS team
python-numexpr-1.4.1-3.puias6.x86_64.rpm	nwcsaf_aapp-7_6-1.x86_64.rpm
python-numpy-1.6.1-17.1.x86_64.rpm	rttov-11.1-1.x86_64.rpm
python-numpy-devel-1.6.1-17.1.x86_64.rpm	pyresample-0.7.13-1.noarch.rpm
scipy-0.10.1-1.x86_64.rpm	pyproj-1.9.3-20130417_1.x86_64.rpm
grib_api-1.9.18-1.x86_64.rpm	pygrib-1.9.6-1.fc17.x86_64.rpm
grib_api-devel-1.9.18-1.x86_64.rpm	proj-4.8.0-4.1.x86_64.rpm
hdf5-1.8.5.patch1-7.el6.x86_64.rpm	proj-devel-4.8.0-4.1.x86_64.rpm
hdf5-devel-1.8.5.patch1-7.el6.x86_64.rpm	proj-epsg-4.8.0-4.1.x86_64.rpm
netCDF4-1.0.9-1.x86_64.rpm	proj-nad-4.8.0-4.1.x86_64.rpm
netcdf-4.1.1-3.el6.5.x86_64.rpm	hlhdf-0.8.1+git20130416-2.x86_64.rpm
	h5py-2.1.2-1.x86_64.rpm

The package nwcsaf_aapp-7_6-1 is version 7.6 of AAPP from the NWPSAF compiled without MAIA or any NWP support. See notes below!

The rttov-11.1 is RTTOV 11.1 from the NWPSAF built to support PPS.

5.3.1.1 Notes

CentOS 6.4 comes with python 2.6.6 and numpy 1.4. Numpy 1.4 is now more than 4 years old and simply too old for PPS. Also CentOS 6.4 doesn't provide grib-api as the corresponding RHEL release does. However, the two grip-api packages can be downloaded from ECMWF.

Also an appropriate version of PROJ.4 was not found at built time, requiring a separate built for that. All that together required the rather extended list of packages listed above to be created by the NWCSAF/PPS team.

The hdf5, hdf5-devel and netcdf packages are normally available in the CentOS *epel* repository and could be installed over the internet provided the rpm package `epel-release-6-8.noarch.rpm` has been installed. We have, however, for convenience and safety decided to include those packages directly here, since these are the exact versions used when we built netCDF4, h5py and hlhdf.

The AAPP package is built only in order to be able to run PPS on global Metop data in EPS level-1 format! It has been compiled without MAIA support, and may not be sufficient for your local processing. So please do not use this AAPP installation for any direct readout processing!

5.3.2 SUSE SLES 11

NB! The rpm packages to be downloaded are all gathered in one big tar file!

The following RPM packages for PPSv2014 are available:

- nwcsaf_aapp-7_6-1.x86_64.rpm
- nwcsaf_ahamap-v2012+git20130418-2.x86_64.rpm
- pps_nwp-0.4.9_29_ga009-1.el6.noarch.rpm
- TBD-02

In addition these 3rd party rpm packages specifically needed by PPS are included in the package tar file (the exact list and filenames and version numbers are TBD):

From the Suse SLES 11 repositories or ready built elsewhere	Built by the PPS team
grib_api-1.9.18-1.x86_64.rpm	netCDF4-1.0.9-20140626_1.x86_64.rpm
grib_api-devel-1.9.18-1.x86_64.rpm	netcdf-4.2.1-15.1.x86_64.rpm
libproj0-4.7.0-5.1.x86_64.rpm	netcdf-devel-4.2.1-15.1.x86_64.rpm
libproj-devel-4.7.0-9.1.x86_64.rpm	netcdf-devel-static-4.2.1-15.1.x86_64.rpm
proj4-4.7.0-5.1.x86_64.rpm	netcdf-doc-4.2.1-15.1.x86_64.rpm
libjasper-devel-1.900.1-134.1.x86_64.rpm	libnetcdf7-4.2.1-15.1.x86_64.rpm
libjpeg-devel-6.2.0-879.3.x86_64.rpm	pyresample-0.8.0-1.fc17.noarch.rpm
python-imaging-1.1.6-168.1.x86_64.rpm	pygrib-1.9.6-1.sles11.x86_64.rpm
	pyproj-1.9.3-20130417_1.x86_64.rpm
	h5py-2.1.2-1.x86_64.rpm
	hlhdf-0.8.1+git20130416-2.x86_64.rpm
	ordereddict-1.1-1.noarch.rpm
	python-numpy-1.6.2-28.1.x86_64.rpm
	python-numpy-devel-1.6.2-28.1.x86_64.rpm

	scipy-0.10.1-1.x86_64.rpm
	rttov-11.1-1.x86_64.rpm

5.3.3 Open Suse TBD

TBD-02

5.3.4 RHEL 6.5

TBD-02

5.3.5 Ubuntu TBD

TBD-02

6. INSTALLING THE PPS-PACKAGE USING SOURCE CODE

PPS may be built from the source code. This may e.g. be useful if you want to have full control over where the PPS is installed and how it is compiled, or if there is no binary installation package available yet for your platform.

The installation from source code requires that you already have all the third party software in place.

If you are on a platform where no binary distribution of PPS and its third party dependencies are available yet, you may need to built and install the third party software yourself from source. ANNEX B provides some guidelines which may be helpful when compiling the third party software.

NB! The NWCSAF/PPS team, however, recommends to first trying to rebuild the binary packages from the available source packages for one of the provided distributions. It is likely that a rebuilt package will be just fine for your OS and platform, provided it does not deviate significantly from one of the provided distributions. Plenty of guidelines on how to rebuild an rpm or debian package from its corresponding source package exist on the internet.

6.1 BUILD AND INSTALL

This section describes the procedure for the installation of the ACPG, AHAMAP and CPP applications, and gives an overview of the final installation. This procedure has the objective to successfully build and install the PGEs and the common library functions (including the library functions to read and remap AAPP output - AHAMAP) into the *install directory*. First we list the necessary pre-requisites, the requirements on the local environment, then how to build and install, and finally the content of a final installation, in terms of directory structure and content.

6.1.1 Before you start building

Prior to building the software as described in the next section, there are a few important things to note:

- The v2014 of PPS will give output in a different format than PPS v2012. See 1.1 for a short description.
- Make sure to use the GNU-*make* and not any possible native *make*! Easily checked by:

```
%> make --version
GNU Make version 3.76.1, by Richard Stallman and Roland McGrath.
Copyright (C) 1988, 89, 90, 91, 92, 93, 94, 95, 96, 97
        Free Software Foundation, Inc.
This is free software; see the source for copying conditions.
There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE.

Report bugs to <bug-gnu-utils@prep.ai.mit.edu>.
```

If you happen to find the native *make* first, do something like this:

```
%> alias make /usr/local/bin/make
```

provided you have your GNU-*make* under `/usr/local/bin`

- Check that you are using appropriate C- and Fortran 90 compilers and linkers. The configure script should hopefully find a suitable one for you, unless they are located under an awkward name or in an awkward location. The configure script searches for Fortran compilers using the following priority list: [f90 pgf90 ifort ifc gfortran epcf90 f90 f77 pgf77 g77]. Likewise it searches for a C compiler according to the following: [cc CC /usr/ucb/cc gcc].

  If you don't have any of these compiler names defined in your system you are still able to make configure find your desired C and Fortran 90 compilers setting environment variables F77/FC and CC or cc. In tc-Shell it may look like this:

  %> `setenv CC` *your-c-compiler*

  %> `setenv F77` *your-fortran-compiler*

  %> `setenv FC` *your-fortran-compiler*

6.1.2 Installing required python packages

The Python parts of AHAMAP and ACPG depend on NumPy at build- and runtime. In addition, ACPG has a few other Python runtime dependencies. These all need to be installed and available on the Python search path (e.g. in the PYTHONPATH environment variable) during configuring, building, and running AHAMAP/ACPG.

You should install the required python packages by yourself (maybe you already have them), and then also make sure that you install the pps_nwp package.

6.1.2.1 pps_nwp

pps_nwp is a python package specially created for the NWC/PPS needs, and which is delivered together with the NWC/PPS software.

Installing the package is installed like this:

```
python setup.py install
```

For a non-default location:

```
python setup.py install --prefix=/usr/local
```

Unit tests can be run by:

```
python setup.py test
```

For more options, see:

```
python setup.py --help
```

6.1.2.2 NumPy

AHAMAP, ACPG, and several of the third party Python packages (e.g. HLHDF, SciPy, and pygrib) require that the same version of NumPy is used at both build- and runtime. This is because they build C-modules which are dynamically linked to the NumPy shared libraries. Make sure you have the intended NumPy version installed and importable when you build these packages and run PPS.

6.1.2.3 Environment variables

Some packages require environment variables to be set before building.

6.1.2.3.1 pygrib

GRIBAPI_DIR

installation directory of GRIB API. Must be set if pygrib is not yet installed, and GRIB API was installed to a non-standard directory.

JASPER_DIR, OPENJPEG_DIR, PNG_DIR, ZLIB_DIR

installation directory of JasPer, OpenJPEG, PNG, and zlib, respectively. Corresponding environment variable must be set if GRIB API was built with support for any of these, regardless of whether the library was installed in a standard or non-standard directory (see http://pygrib.googlecode.com/svn/trunk/docs/index.html).

6.1.2.3.2 h5py

HDF5_DIR

installation directory of HDF5. Must be set if h5py is not yet installed, and HDF5 was installed to a non-standard directory (see http://h5py.alfven.org/docs-2.0/intro/build.html).

6.1.3 Installing AHAMAP

The AHAMAP package come as a gnu-zipped tar-ball and needs to be unpacked before the installation process can begin. For instance, standing in the directory where you want the AHAMAP source code to reside, do something like this (provided your tar command supports the z option):

```
%> tar xvzf ahamap_v XXX.tgz
```

(where *XXX* identifies the version). The installation of ahamap is quite straight forward, basically it should just be to run the configure script, compile the code and then install the package.

The configuration script will try to determine where specific include files and libraries are located. This script also provides some optional configurable parameters. The recommended configuration looks like this (of course with the appropriate installation paths for your local environment substituted wherever relevant):

```
%>./configure –prefix=/usr/local/AHAMAP –with-
proj=/usr/local/include,/usr/local/lib
--without-aapp
```

The above syntax should be well known to most Unix-administrators, as the `autoconf` configuration script is commonly used in free software distributions. It is possible to specify many more options to the `configure`-script, than what is indicated above, but it will normally suffice to specify only a few, as shown above. All possible options can be listed using the `help` flag:

```
%> ./configure --help
```

The output of this command is lengthy and is not shown here. Try it yourself.

Compiling AHAMAP is the next step. This is done by just typing `make` in the top level directory of the source-distribution:

```
%> make
```

Before installation it is time to test if the package was properly built:

```
    %> make check
```

Finally, ahamap should be installed. This is done by typing:

```
    %> make install
```

For more information on how to configure and install AHAMAP, see [RD.14].

NB! Note that we here build AHAMAP without support for AAPP. From v2014 the AAPP libraries are not needed anymore.

6.1.4 ACPG: `Configure`, `make`, `make install`

The building and installation of the ACPG, which is written mainly in C and python, are alleviated using a `configure` script, generated using the Gnu `autoconf` application. The `configure` script automatically determines the platform, compilers, necessary compiler options, etc., and finds the necessary packages needed to build the PPS-software. All the user has to do, is to provide the script with the information on the location of a few essential (third party) software packages and where to install the application. This is done via command-line options/flags, like in the example below:

```
%> ./configure --prefix=/usr/local/ACPG
--with-hlhdf=/usr/local/HLHDF
--without-aapp
--with-ahamap=/usr/local/AHAMAP
--with-rttov=/usr/local/RTTOV11_1
--with-proj=/usr/local/include,/usr/local/lib64
--datadir=/data/SAFNWC_PPS
--with-gac=no
```

`--prefix` is the path where to put the ACPG installation.

`--datadir` is the path where to put the in and out data, see 6.1.7.

`--with-gac` decides if GAC-settings are used or not.

All possible options can be listed using the `help` flag:

```
%> ./configure --help
```

The output of this is not shown here, but as seen from the available options it is possible to configure the installation so as not to include different parts of the PPS package. For instance, it is possible to deselect support for generating the CTTH (using `--without-ctth` or `--with-ctth=no`) and the RTTOV package needed by CTTH (using `--without-rttov` or `--with-rttov=no`) or the generation of the Precipitating Clouds (using `--without-precip` or `--with-precip=no`). Since the Cloud Mask and Type are needed by the other products these products are mandatory for an installation. It is also possible to build acpg without aapp support using the option –without-aapp. This means that sun and satellite angles cannot be generated before the satellite reception. Instead these angles have to be derived using the information in the AVHRR level 1b file.

It is of course strongly recommended to build the full acpg package with all possible support as shown in the example above. The exception being AAPP; v2014 and later can be build without AAPP support which ease the building process somewhat.

The `configure` script is delivered as part of the application. Thus, the user does not need to run `autoconf` herself to generate the `configure` script.

Running the configure-script takes input from a file called `def.mk.in` and generates a file `def.mk`, which is subsequently used as input to all *makefiles* in the application. The configure-script also creates the file `soure_me`, taking input from `source_me.in`. Read more about the use of `source_me` in UM/2 [RD.2]

Compiling the ACPG is the next step. This is done by just typing `make` in the top level directory of the source-distribution (where among other files the `configure` script and the `def.mk` file resides):

```
%> make
```

During this step every sub-directory of the distribution will be checked and all necessary code compiled and linked (if it wasn't successfully compiled in a previous step). Static and dynamic libraries and binary executables will be generated and stored in their respective sub-directories. It is not until the installation step (below) that the code is copied to the install directory.

Before installation it is time to test if the package was properly built.

```
    %> make check
```

This test is a rather rudimentary one. It goes through all the sub-directories where python shared libraries should exist. It checks if the expected library is available and checks if it is possible to successfully import it into python. If the shared library is not available or it cannot be imported something is wrong. It might be, that one of third party softwares is not properly built, or that some compiler options are missing or wrong.

The test can be regarded as a spot test to see if anything is seriously wrong or not. A successful result of the above step gives an indication that the package is properly built, but there is no guarantee.

Now it is time to install the package:

```
%> make install
```

This step will copy all necessary executables, include files, (static and dynamic) libraries, configuration files, and static data (algorithm tables etc) to the install directory specified with the `--prefix` flag.

Note: If any of the sub-directories are missing in the distribution, the make-command may enter a loop. Make sure that all directories do exist.

NB! If you intend to run PGE05, you also need to install CPP before having the full functionality on place.

Before the installation can be considered complete, we need to create the directory tree to hold the input, output, and intermediate data files, and to install the test data and the architecture independent files not included among the static data and configuration files handled in the above step. This will be done like this:

```
%> make install-data
```

The above step will install the test data, various auxiliary data (land cover, elevation, surface emissivity etc) and a set of physiography (*landuse*, *fraction of land* and *elevation*) data files on some default regions, in the directory given by the `--datadir` flag to the configuration script. If this was not specified in the above, the data-directory will reside under PREFIX`/share` which is probably not where you want it. So please specify the desired location yourself!

The directory structure obtained after installation is described in 6.1.7.

6.1.5 Test data and static data

A test dataset with necessary input data to run the ACPG is provided as part of the PPS-package. The test data are installed in the appropriate sub-directories under the *install-datadir* directory tree. During delivery of the PPS-software also a *reference dataset* is provided. The reference dataset contain intermediate and output data generated when running the software at SMHI on the test dataset.

Please observe that the test dataset is not only test data, but also contain static data (eg. physiography data) actually needed for running ACPG.

To be able to run CPP you need the CPP ANC-data, delivered in a separate package. Those data should be placed in, or linked to, `import/CPP_data/source`. This directory has to be created manually.

6.1.6 CPP installation

The CPP package comes as a gnu-zipped tar-ball and needs to be unpacked before the installation process can begin. For instance, standing in the directory where you want the CPP source code to reside, do something like this (provided your tar command supports the z option):

```
%> tar xvzf cpp-vXXX.tgz
```

(where *XXX* identifies the version). The installation of CPP is quite straight forward, basically it should just be to run the configure script, compile the code and then install the package.

The configuration script will try to determine where specific include files and libraries are located. This script also provides some optional configurable parameters. The recommended configuration looks like this (of course with the appropriate installation paths for your local environment substituted wherever relevant):

```
%>./configure –-prefix=/usr/local/ACPG
–-with-hlhdf=/usr/local/HLHDF
–-datadir=/data/SAFNWC_PPS
--disable-msg
```

Please observe that the `--prefix` should be the same as for ACPG, for being able to run the CPP the intended way. As well as the datadir should be the same as for ACPG, to be able to get input for CPP from the ACPG processing.

`--disable-msg` should be set, as the CPP delivered by SAFNWC-PPS is not supposed to process MSG-data. (Though there might be MSG-processing parts in the code, as it is shared with CM-SAF, processing both PPS and MSG data.)

All possible options can be listed using the `help` flag:

```
%> ./configure --help
```

Compiling the ACPG is the next step. This is done by just typing `make` in the top level directory of the source-distribution:

```
%> make
```

Finally, cpp should be installed. This is done by typing:

```
%> make install
```

For cpp you also need some ancillary data. The file with those data should be unpacked in your data directory under the sub-directory `import/CPP_data/source`.

6.1.7 ACPG installation directory structure

The ACPG will run in the directory structure as shown in Figure 2. The root directory `/install-prefix` of the ACPG will be defined by the user during building and installation. The ACPG programs, header files, libraries, scripts, configuration files, and static data (thresholds etc) will be found in sub-directories hanging from this root directory. Data will be found under the `install-datadir` directory tree. Global physiography data will be installed in the `import/AUX_data/source` and regional physiography data will be generated upon your request and put under `import/AUX_data/remapped`. The install procedure will install one pre-defined region.

The content of the directories presented in Figure 2 and Figure 3 is explained in the following sections.

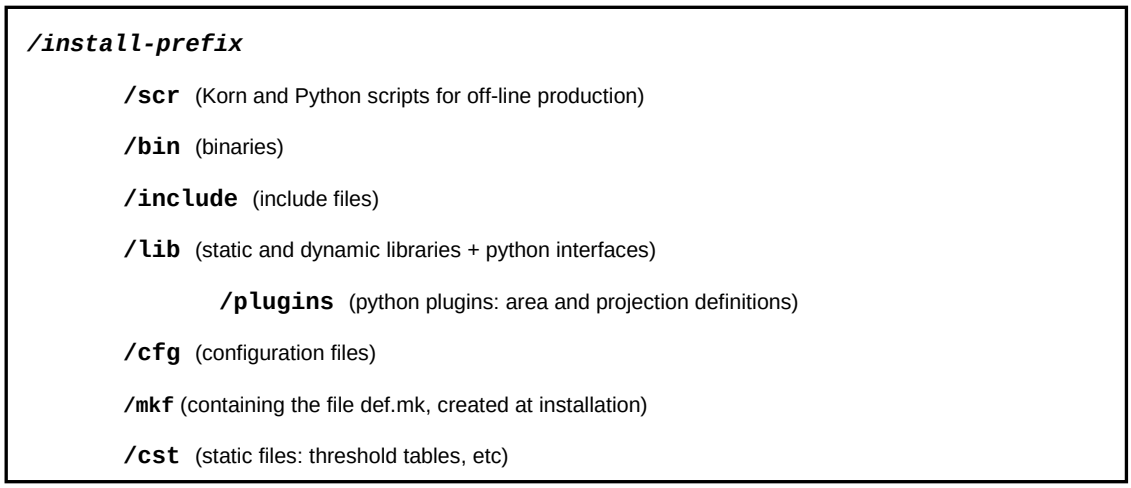

Figure 2: PPS-package installation directory structure after installation of the ACPG package

/install-datadir

/log

/import (input data files and temporary data files)

/ANC_data (Ancillary + temporary data)

/remapped (angles & threshold files on satellite swath –hdf5)

/NWP_data (NWP data)

/source (Grib data)

/remapped (fields remapped on satellite swath –hdf5)

/ICEMAP_data (OSISAF data)

/source (from OSISAF, Grib data or netcdf)

/remapped (fields remapped on satellite swath – hdf5)

/PPS_data (Satellite data)

/source (AAPP output)

/remapped (remapped AAPP data satellite swath – hdf5)

/EMISS_data (Emissivity data)

/source (emissivity tiles –hdf5)

/remapped (remapped emissivity data on satellite swath – hdf5)

/AUX_data

/source (global physiography)

/remapped (physiography files on region or satellite swath – hdf5)

/export (output)

Figure 3: PPS-package data directory structure after installation of the ACPG package

6.1.7.1 Directory "scr"

This directory contains Python scripts for off-line production of the PPS products and for testing the installation of the PPS package. If you choose to operate the PPS-package, using either a crontab-solution type scheduler or your own local scheduler, you could interface to the python scripts available here. You also find a few Korn shell scripts here.

6.1.7.2 Directory "bin"

This directory contains all the executable code of the PPS package. The install script of the PPS package stores the latest binary installation using the prefix "OLD" on all files before installing. In this way it is always possible to go back to the previous version.

If the user wishes to keep a full track of all previous installations it may be convenient to save each version in this directory, creating sub-directories for each operative version. The executables of the PPS-package are the online and pre-processing parts of the PGEs, plus a few test programs.

6.1.7.3 Directory "include"

All C-header files, with extension `h`, of the PPS-package will be installed in this directory. This concerns header files of the PGEs and the common functions.

6.1.7.4 Directory "lib"

All static and dynamic (shared) libraries, with extension `a` and `so`, will be stored in this directory, together with python modules and interface code.

The static libraries each contain a collection of common or PGE-specific functions, implemented in C. The libraries of common functions are used already before installation when building the online and pre-processing parts of the PGEs. They are only stored here for convenience, and could be applicable to the user if some functionality of the PPS-package is desired in other applications. This also applies to the static libraries collecting the PGE-specific functions. An example of the usage of those libraries are when validating the Cloud Mask against SYNOP data, where first the Cloud Mask algorithm is run on the so-called MSMS database using a specific interface between the database and the Cloud Mask.

The shared libraries each contain a collection of Applications Programming Interfaces (API's) implemented in C for the Python programming language, allowing to access functionality in the PPS-package from Python. In this case it is mostly I/O routines making it possible to read and write the PPS output products and intermediate data files. A few python modules, with extension `py`, to ease the use of the API's are also included in this directory.

6.1.7.5 Sub-directory "lib/plugins"

Python plugins containing area and projection definitions for the regions installed (corresponding to the physiography files installed under `import/AUX_data/remapped`) are found under the `plugins` sub-directory under `lib`. Be sure to include the `lib` sub-directory in your `PYTHONPATH`.

6.1.7.6 Directory "cfg"

All the configuration files will be stored in this directory. This will enable all programs to access the configuration parameters needed for their execution.

In Table 3 a list of all configuration files in this directory is provided.

Description	Filename
Environment variables. Basic PPS definitions (directory paths, file names, etc.).	`.profile_pps`
NWP Grib Configuration File (optional. If not available, default settings will be used.)	`pps_nwp.ini`
Region Configuration File	`region_config.cfg`
Python script configuration, including some PGE03 and PGE05 configurations.	`pps_basic_configure.py`
Communication configuration	`pps_config_common.cfg`
PGE 01-03 Model Configuration File	`pps_config_common.cfg`
PGE 04 Model Configuration File	`generate_precipitation.cfg`
Sea surface temperature coefficients. Used for PGE 01.	*satellite*`_sst.ini` (for all satellites)

Other Precipitating Clouds specific files	`avhrr_PI_day_`*X*`_alg_`*Y*`.dat`
	`avhrr_PI_night_alg_`*Y*`.dat`
	(X=3a or 3b, Y=0,1,2,3 or 4)
	`land_scat_clas.dat`
	`coast_scat_clas.dat`
	`sea_scat_clas.dat`
	`backgilbcoeff.hdf`
	`wvpcoeffs.hdf`
	`lwpcoeffs.hdf`
	`t89coeffs.hdf`
	`amsu_landsea_mask.hdf`

Table 3: Configuration files in the `/cfg` directory.

6.1.7.7 Directory "cst"

`cst` is an abbreviation for, or contraction of, the word *constant*. All static tables or data files used by the PGEs are stored in this directory, except for the global and regional physiography data. These files are either stored in ASCII format or using the HDF5 file format. The satellite specific threshold tables for the Cloud Mask and Type, and the corresponding static threshold offsets defined from tuning activities are examples of such files.

When a new EPS or NOAA satellite is launched, new files, to be generated and delivered by SMHI, should be added to this directory.

The type of files in this directory differs from the configuration files in the `cfg` directory. The configuration files have mostly something to do with configuring the software. Only a few of the configuration files have implications to the (scientific) algorithms, and then changes by the user could be motivated. The files in the `cst` directory are specific to the algorithms and have been generated during careful development, e.g. from RTM calculations, and should normally not be changed by the user.

6.1.7.8 Directory "tmp"

Temporary files, not stored in some of the subdirectories to `import`, described below, will be stored here. This directory is chosen by the user and can be configured using the `TMP_DIR` parameter in the `.profile_pps` file. Cleaning of this directory will be necessary in operational service as content will be added continuously. This can be accomplished automatically using scripts available from the ACPG-package (see UM/2 [RD.2]).

6.1.7.9 Directory "import"

All input files needed by the SAFNWC/PPS application (except for SAFNWC/PPS output products) will be linked to this directory, or stored directly in it. This enables all the PGEs to find the files they need.

An overview of the import directory and its sub-directories is provided in Figure 3. More details can be found in ICD1 [RD.4].

The `import` directory contains seven sub-directories. These sub-directories usually have two sub-directories each, named `remapped` and `source`. Data which have been re-mapped to the present satellite projection shall be found under `remapped`, and data in their original format, that is as they are provided by the user according to the specified external interfaces, are stored under `source`. The user of the SAFNWC/PPS application will be in charge of filling the `source` directories under `PPS_data` and `NWP_data` with the adequate files, as well as `ICEMAP_data` –though they are optional. Data for `AUX_data/source`, `EMISS_data/source` and `CPP_data/source` comes with the PPS-package.

It is also possible to use the environment variables to define other directories, than this default structure, for input and intermediate files.

6.1.7.10 Directory "export"

All products generated by the SAFNWC/PPS application will be saved to the `export` directory with the naming convention described in DOF [RD.11] and UM/2 [RD.2]. The user will then find all the products there.

Cleaning of this directory will be necessary in operational service as content will be added continuously. This can be accomplished automatically using scripts available from the ACPG-package (see 'Automatic cleaning' in UM/2 [RD.2]). The maximum amount of disk space occupied and number of days to keep files should (and can) be decided by the user. Archiving to another directory or a tape is expected to be handled by the user and is considered outside the scope of the PPS-package.

6.2 BUILD AND INSTALL, AGAIN…

6.2.1 ACPG, CPP & AHAMAP

In theory it should be necessary only to follow the building and installation procedure described above once for your local platform after having acquired the release, and you will be having a fully functional SAFNWC PPS-system. And repeating it should be relevant only when getting a new full release of the PPS-software or when getting patch releases. However, reality is often somewhat different and more complicated.

For instance it is likely that you have missed to specify one or more of the configuration flags correctly, or you simply change your mind on how to configure your installation. In addition, the PPS-software may not be fully prepared for your environment, and you might have to make changes to makefiles or other building dependent files in order to get a version that builds on your system. Thus, in practice, before you have a working installation you will have invoked the configuration and make commands several times.

Whether you are going to run the building and installation procedure again after having corrected a few things, or whether you do it again after having acquired updates (patches) or a full new release, there are a few things you should be aware of:

- If the `./configure` (as described in section 6.1.4) fails or you want to change the configuration flags and run the configuration again, make sure to clean up the files generated during the configuration. If `./configure` was terminated successfully it suffices to do a `make distclean`. This also cleans all the sub-directories for already generated objects and libraries. If `./configure` didn't terminate successfully you may have to delete some of the following files: `config.log`, `config.status`, `source_me` and `def.mk`.
- If you have got an update or a new release, it is a good idea to do a `make distclean` in the top level directory before start, and then run `./configure`, `make`, etc, as described above.
- Installing on top of a previous installation is handled automatically using `make install` and `make install-data` (only applicable to ACPG). However, it is important to notice that `make install-data` does not overwrite already installed data. Thus if these data files have been updated (from one release to another), you will have to clean up the data under the installation before making a new installation. Normally, however, the data files handled with the install-data flag will not be updated very frequently.

6.3 ERROR MESSAGES AND RECOVERY PROCEDURES

As mentioned earlier in section 6.1.1 it is vital to use the correct compilers and linkers, and to use Gnu-make rather than any type of native make-facility. But at the same time it is very often that the problems in building the software originate from not using the right applications.

6.3.1 Configure problems

TBD-03

6.3.2 Make-problems

TBD-03

6.4 SUMMARY OF PROBLEMS AND HOW TO FIX THEM

TBD-03

7. SOFTWARE OVERVIEW

The SAFNWC/PPS software package, including test data and reference data, actually comes in five pieces or sub-packages:

1. **py-packages**: A package containing python packages used by PPS. Most of them (eg. Numpy, h5py) can be found by other means, if preferred. While a few (so far only pps_nwp) is PPS specific and is only delivered this way.
2. **AHAMAP**: The AMSU-HIRS-AVHRR extraction and re-mapping tool
3. **ACPG**: This is the main software of PPS. The AVHRR/AMSU Cloud Product Generation software, including functions for extraction and re-mapping of input and auxiliary data.
4. **CPP**: The Cloud Physical Properties generation software. (Developed by CM-SAF)
5. **PPS reference dataset**: Intermediate and final output of the PPS-package, generated by running the installation at SMHI, on the test dataset included in the ACPG-package.

Each sub-package comes as a *gzip compressed* (compressed) *tar* file, containing source code, configuration files, static data files and/or test data files. As the ACPG-package contains large static data files and test data, this package will usually be split into two or more tar-balls. When upgrading the PPS package it will often suffice to only download the source code, as the static data and test data are less likely to change between versions of the PPS-package.

The reference dataset is available for you to check against your locally produced output generated from the accompanying test dataset. There is a small python program `pps_compare_outputs.py` under `/pythonlib` (see further down) which may be a starting point when wanting to make more quantitative statistics on the possible differences.

The building and installation of the packages should preferably be done according to the order indicated above. AHAMAP is required by, and thus must be installed prior to, the ACPG package. The CPP package is supposed to be installed in the same place as the ACPG package is installed.

All source code created by the PPS software developers are saved on a version handling tool (GIT) and more traceability information can be made available to users by request sent through the SAFNWC Helpdesk.

7.1 AHAMAP

The AHAMAP software is a collection of C and Fortran (two files only) routines and Python interfaces to read NOAA/Metop satellite data in level 1 file format, and re-map these data to a user-defined map-projection. The AHAMAP software performs the calibration and navigation of the satellite data, using only the calibration and navigation information appended in the level 1 data files.

The AHAMAP software provides the user with an easy way of projecting the satellite data to a common map-projection surface, for example using a polar stereographic projection.

7.1.1 Navigation without support for AAPP

Both ACPG and AHAMAP should be built without AAPP support (see section 6.1.3), the AAPP functionality is not needed since the navigation of AVHRR data will be done using extensive interpolation/extrapolation on the information already available in the level 1b file.

In order to run PPS on global Metop GDS granules the AAPP library must be installed, however. PPS uses an AAPP file format converter to convert the EPS level1 data to AAPP level1b format. All which is needed is that the environment variable AAPP_PREFIX is set correctly. Here is what the acpg configure tells you when building without AAPP support:

```
checking hardware platform... x86_64
checking for python... yes
checking for python... /usr/bin/python
Version of python = 2.6 > 2.5
checking for the compiler used for generating python...
checking for /usr/lib/python2.6/config/Makefile... no
checking for /usr/lib64/python2.6/config/Makefile... yes
Searching for C compiler in /usr/lib64/python2.6/config/Makefile
Found python compiler: gcc
Version of grib_api used by pygrib: 1.9.9 <= 1.9.9
Setting F77=gfortran, needed for acpg
checking for Fortran 77 compiler default output file name... a.out
checking whether the Fortran 77 compiler works... yes
checking whether we are cross compiling... no
checking for suffix of executables...
checking for suffix of object files... o
checking whether we are using the GNU Fortran 77 compiler... yes
checking whether gfortran accepts -g... yes
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking how to run the C preprocessor... gcc -E
checking for ranlib... ranlib
checking for ar... ar
checking projects.h usability... yes
checking projects.h presence... yes
checking for projects.h... yes
checking for pj_inv in -lproj... yes
checking If it is old or new version of proj... New version
"HLHDF supports debug routing features"
checking aapp_c_funcs.h usability... yes
checking aapp_c_funcs.h presence... yes
checking for aapp_c_funcs.h... yes
checking for ellipse in -laapp... yes
checking for AHAMAP version... v2012-patch20120927
checking for numpy>=1.4.1... 1.6.2
checking for scipy>=0.8... 0.10.0
checking for h5py>=1.4... 2.0.1
checking for pps_nwp>=0.4.7... 0.4.9-3-g397c
checking for netCDF4>=0.9.4... 1.0.2
checking for pyresample>=0.7.9... 1.1.0
checking system version... Linux-2.6.32-431.20.3.el6.x86_64
checking Linker Options for gfortran on Linux-2.6.32-431.20.3.el6.x86_64 with -
L/usr/lib64  -lgfortran... Yes
checking for library containing socket... none required
checking for library containing gethostbyname... none required
checking for socket... yes
checking for gethostbyname... yes
checking for select... yes
checking for AAPP lib... suppressed
AAPP library suppressed. AAPP is required for processing GlobalMetop
GlobalMetop data can still be processed if AAPP_PREFIX is set and
convert_avh1b.exe is on PATH
checking Fortran Linker options (-Xlinker -Bstatic -Xlinker -Bdynamic)... "Yes"
Setting LD_FORCE_STATIC =  -Xlinker -Bstatic
Setting LD_FORCE_SHARE = -Xlinker -Bdynamic
```

```
checking for "/local_disk/opt/RTTOV11/11_1/mod/rttov_coef_io_mod.mod"... yes
checking for "/local_disk/opt/RTTOV11/11_1/mod/rttov_const.mod"... yes
checking for
"/local_disk/opt/RTTOV11/11_1/include/rttov_dealloc_coef.interface"... yes
checking for rttov_init_rad in -lrttov11.1.0_main... yes
checking for gac settings... GAC is not used
checking for the Generic Mapping Tool (GMT) used to generate geographical
plots... /usr
checking for GMT version... Unknown
configure: creating ./config.status
config.status: creating ./common/pyutils/test_ch3b_refl.py
config.status: creating ./def.mk
config.status: creating ./source_me
config.status: creating ./cfg/.profile_pps
config.status: creating ./test/.profile_pps.in
```

The lines referring to the missing AAPP support are highlighted in green.

7.1.2 Mixing C and Fortran

TBD-04

7.2 ACPG

Here we will concentrate on the description of the core part of the software, namely the ACPG. It includes, among other things, the algorithms for PGE01-PGE04, but not for PGE05. PGE05 is found in CPP, see 7.3.

The ACPG includes the following (though packed in two tar balls):

- Source code
- Configuration files (with exemplified settings)
- USGS land cover characterisation data on a continental basis, covering all the earth – native format (Azimuthal Equal Area projection)
- USGS Gtopo 30 elevation and roughness database on many small tiles covering all the earth – HDF5 (Plate Caree projection)
- Fraction of land data derived from the USGS Global 1km Land Cover Characterization database –HDF5 format. One file per continent of the earth.
- Land surface emissivity data, on tiles covering the entire earth, for all months of the year (a climatology from the years 2003-102, calculated from the standard MODIS product MYD11C3) – HDF5.
- Landuse, fraction of land and elevation data (physiography data) prepared for one default region
- Other static data files (e.g. thresholds derived from RTM calculations)
- A test data set (including AVHRR/AMSU level 1b/1c data and NWP Grib)

7.2.1 Source distribution directory structure

The source distribution of the ACPG part of the PPS package is delivered as a compressed (`gzip compressed`) `tar` file. Here follows a short overview of the structure and content of the acpg-source distribution when unpacked.

The acpg- source directory contains a number of subdirectories with the source code. `/cfg` contains configuration files. `/cst` contains threshold tables, and auxiliary data. In `/test` you find scripts for regression test. The source distribution also contains a full test data set, which is found under the `/testdata` sub-directory.

An overview of the main structure and content of the acpg source directory is shown in Table 4.

When installing the PPS package, an ACPG- install directory structure is created where three of the sub-directories are to some extent identical to the `/scr`, `/cst` and `/cfg` sub-directories of the source directory described here.

ACPG		Parent directory for the SAFNWC/PPS source distribution
	`cfg`	Configuration files
	`testdata`	Directory with SAFNWC/PPS input files – for testing only.
	`preproc_cmask_ctype`	Cloud Mask and Type pre-processing module – source code
	`cloudmask`	Cloud Mask – PGE01 – source code
	`cloudtype`	Cloud Type – PGE02 – source code
	`ctth`	CTTH – PGE03 – pre-processing and online processing modules – source code
	`precipclouds`	Precipitating Clouds – PGE04 – pre-processing and online processing modules – source code
	`common`	Common libraries – miscellaneous common functions including I/O handling – source code
	`pyutils`	C-API for Python interface to the region_config.cfg file and the log-writer (pps_logmanager.c)
	`FullSwath`	Python scripts to run PGE:s in satellite projection (direct read-out and Global Metop)
	`GlobalMetop`	Some functionality to run PPS on Global Metop data
	`GAC`	Python scripts to run PPS on GAC data.
	`scr`	Korn shell and Python scripts
	`pythonlib`	Miscellaneous Python functions
	`plugins`	Default projection and region definitions (pcs and area plugins) for python-Proj interface
	`pyepshdf`	Python API's for reading and writing products and intermediate data files in HDF5
	`satproj`	C-code with Python API's for mapping data between Proj map projections and satellite projection.
	`usgs2region`	Generation of physiography data on region extracting and re-mapping the global USGS landuse and topography data
	`cst`	Static auxiliary data (including PGE specific thresholds etc.)
	`USGS_FracOfLand`	hdf files with fraction of land data derived from the USGS Global 1km Land Cover Characterization database. One file per continent of the earth.
	`USGS_Landuse`	*g*-zipped global USGS land cover characterisation data on a continental basis
	`USGS_Topo`	*g*-zipped global USGS Topography data (GTOPO 30 Arc Seconds database) in large tiles
	`Emissivity`	g-zipped hdf files with land emissivity data, on tiles covering all the earth, for all month
	`OSISAF_ice`	A file compensating for missing ice concentration data around the pole.
	`test`	Functions for making regression test.

Table 4: Directory structure and content of the SAFNWC PPS ACPG source distribution

7.2.2 Mixing C and Fortran

The ACPG software mixes C and Fortran only in its interface with the RTTOV library. RTTOV is needed for PGE03 to generate TOA clear radiances.

7.3 CPP

The CPP package contains the C-code for generating the Cloud Physical Properites product. This code is developed by CM-SAF.

Python scripts for calling the C-program, for generating images etc. are found in the ACPG-package. Also, there is no specific test data set for CPP, as the test data of ACPG can be used.

The CPP includes the following (though packed in two tar balls):

- Source code
- Ancillary data

Downloading/installation of the CPP package is needed only if the CPP product (pge05) is to be generated, otherwise the CPP package can be omitted.

7.3.1 Source distribution directory structure

The source distribution of the CPP part of the PPS package is delivered as compressed (`gzip compressed`) `tar` file. All files are in one directory.

When installing the CPP package, an install directory structure is created where three sub-directories are created: `/bin`, `/cst` and `/lib`. To have CPP working together with ACPG (eg. using the python scripts in ACPG) the installation should be done on the same place as the ACPG installation –and thus using the same sub-directories as the ACPG installation. CPP and ACPG can be installed in any order.

The compressed tar file, containing CPP ancillary data, is supposed to be unpacked in the data-structure on PPS in the place `import/CPP_data/source`. Or it can be unpacked anywhere and be linked to `import/CPP_data/source.`

8. CONFIGURATIONS

8.1 ENVIRONMENT VARIABLES

The SAFNWC/PPS application will be configured using a number of Unix environment variables plus a set of configuration files (see UM/2 [RD.2]). Whenever a PGE or a pre-processing module is started via a Unix Korn shell script the environment variables should be set for the given process. Similarly, when running in off-line mode, the necessary environment variables must be set in the shell before executing the modules.

The environment variables are set in the dot-file "`/cfg/.profile_pps`". Environment variable may also be set using the `source_me` file. This file is created in the source distribution main directory at installation. The environment variables concerns mainly directory paths, and filenames, etc., but also a few processing options. Some of the variable names, with explanations are given below. Not all variables are described here, but those most relevant for the users. When the table says "Need to be adapted by user", please note that a well chosen configure call, already have set most of them to desired values.

KEYWORD	Description	Type	Possible Value(s)
PPS_SOFTWARE	ACPG install-directory	string	Need to be adapted by user
AHAMAP_SOFTWARE	AHAMAP install directory	string	Need to be adapted by user
DATA_DIR	ACPG data directory	string	Need to be adapted by user
TMP_DIR	Directory to hold temporary files	string	Need to be adapted by user
AAPP_PREFIX	Path to the AAPP installation. In needed when running Global Metop cases, and AHAMAP/ACPG are installed without AAPP-dependence.	string	Sometimes needed to be set by user
PYTHONPATH	The pythonpath. Builds on the existing one.	String	Not necessarily to be changed by user
PATH	The search path for executables.	String	Not necessarily to be changed by user
SM_MISSING_IR_VALUE	The value for missing data in the AVHRR IR channels. Depends on AHAMAP	integer	-32000
SM_LOGGING	Specifies the debug level	Integer	0 (No Debugging messages) to 9 (Full debugging)
SM_LAND_PROC	Indicates whether the Cloud Mask should be derived over land (useful for the OSISAF	Integer	0=False; different from

KEYWORD	Description	Type	Possible Value(s)
	where a cloud mask is needed only over sea)		0 = True
SM_SEA_PROC	Indicates whether the Cloud Mask should be derived over sea (useful for the Land SAF)	Integer	0=False; different from 0 = True
SM_WRITE_NETCDF	Defines whether NetCDF file are written by the PGEs, together with the hdf5-files.	Integer	0=False; 1=True
SM_WRITE_LATLON_IN_NETCDF_CMA SM_WRITE_LATLON_IN_NETCDF_CT SM_WRITE_LATLON_IN_NETCDF_CTTH SM_WRITE_LATLON_IN_NETCDF_PC SM_WRITE_LATLON_IN_NETCDF_CPP	Defines whether the NetCDF file should contain latitude and longitude data. Defined PGE by PGE.	Integer	0=False; 1=True
SM_WITH_ABSOLUTE_AZIMUTH_ANGLES	Indicated whether the sunsatangle file should contain absolute sun- and sat-azimuth angles. (Azimuth difference is always calculated, as well as the sun- and sat- zenith angles.) This option is designed for using with GAC-data.	Integer	0=False; different from 0 = True
SM_OVERWRITE	Specifies if intermediate files (thresholds, nwp-remapped, etc) should be overwritten	Integer	0=False; different from 0 = True
RTTOV_COEFF_DIR	Directory of where to find the RTTOV coefficient file	string	Do not change!
SM_AAPP_DATA_DIR	The data directory of where to find to AAPP level 1b/1c data files. (One sub-directory for each satellite pass)!	string	May be changed by user.
SM_CFG_DIR	Directory path of the cfg directory	String	Ex.: PPS_Installation/cfg
SM_CFG_FILE	The name of the file with common configuration parameters for PGE01-03	string	Do not change!
SM_CST_DIR	Directory path of the cst directory	String	Ex.: PPS_Installation/cst
SM_USGS_DIR	Directory holding the global USGS landuse and Topography data	string	Ex.: PPSdatadir/import/AUX_data/source
SM_AUXILIARY_DIR	Directory holding the re-mapped physiography data	string	Ex.: PPSdatadir/import/AUX_data/remapped
SM_AVHRR_DIR	Directory path of the re-mapped AVHRR data	String	Ex.: PPSdatadir/import/PPS_data/remapped

KEYWORD	Description	Type	Possible Value(s)
SM_SUNSATANGLES_DIR	Directory path of the re-mapped sun and satellite angle files	String	Ex.: PPSdatadir/import/ANC_data/remapped
SM_NWPSOURCE_DIR	Directory path of the NWP data	String	Ex.: PPSdatadir/import/NWP_data/source
SM_NWPDATA_DIR	Directory path of the re-mapped NWP data	String	Ex.: PPSdatadir/import/NWP_data/remapped
SM_ICEMAPSOURCE_DIR	Directory path of the OSISAF ice map data.	String	Ex.: PPSdatadir/import/ICEMAP_data/source
SM_ICEMAPDATA_DIR	Directory path of the re-mapped OSISAF ice map data.	String	Ex.: PPSdatadir/import/ICEMAP_data/remapped
SM_EMISSSOURCE_DIR	Directory path of the emissivity maps.	String	Ex.: PPSdatadir/import/EMISS_data/source/EmissivityTiles
SM_EMISS_DIR	Directory path of the re-mapped emissivity maps.	String	Ex.: PPSdatadir//import/EMISS_data/remapped
SM CPPANC_FILE	Path and name of the CPP ANC-file	String	${DATA_DIR}/import/CPP_data/source/cpp_anc_clim.v*XXX*.h5
SM_CPPLUT_FILE	Path and name of the CPP look-up-table.	String	${DATA_DIR}/import/CPP_data/source/cpp_lut.5chan.16bit.v*XXX*.h5
SM_CPP_CTTH_MANDATORY	1=CTTH is mandatory as input (i.e. stop if missing) to CPP. 0=not mandatory, but use if available	Integer	0=False; 1=True
SM_CPP_SNOWDATA_MANDATORY	1=NWP snow is mandatory as input (i.e. stop if missing) to CPP. 0=not mandatory, but use if available	Integer	0=False; 1=True
SM_CPP_ICEMAP_MANDATORY	1=OSISAF ice map is mandatory as input (i.e. stop if missing) to CPP. 0=not mandatory, but use if available	Integer	0=False; 1=True
PFS_LVL1_DIR	Directory path of the level 1 pfs data.	String	Ex.: PPSdatadir/import/PPS_data/source

KEYWORD	Description	Type	Possible Value(s)
BUFR_LVL1_DIR	Directory path of the level 1 BUFR data.	String	Ex.: PPSdatadir/import/PPS_data/source
SM_THRESHOLD_IMAGE_DIR	Directory path of the regional threshold (for cloud mask and type) data	String	Ex.: PPSdatadir/import/ANC_data/remapped
SM_THROFFSETS_NAME	The name of the file (excl. directory path) holding the threshold offsets used by PGE01. Should be matched with SM_KERNEL_SIZE, i.e. threshold_offsets.cfg for kernel size 5 and threshold_offsets_gac.cfg for kernel size 3.	string	Ex.: threshold_offsets.cfg or threshold_offsets_gac.cfg Do not use other values!
SM_CLOUDTYPE_OFFSETS_NAME	The name of the file (excl. directory path) holding the threshold offsets used by PGE02 (Currently identical as the one for PGE01). Should be matched with SM_KERNEL_SIZE, i.e. threshold_offsets.cfg for kernel size 5 and threshold_offsets_gac.cfg for kernel size 3	string	Ex.: threshold_offsets.cfg or threshold_offsets_gac.cfg Do not use other values!
SM_PSUNGLINT_NAME	Name of the sunglint probability table	string	Ex.: sunglint_probabilities.hdf
SM_CH3B_RADIANCE_TAB_NAME	Name of the file with the table holding the conversion from channel 3b radiance to Tb	string	Ex.: ch3b_radiance_tab
SM_PRODUCT_DIR	Directory path of the PPS output products	String	Ex.: PPSdatadir/export
SM_PRECIPITATION_CONFIG	Name of PGE04 configuration file	String	Ex.: PPS_Installation/cfg/generate_preciptation.cfg
SM_REGION_CONFIGURATION_FILE	Name of the region configuration file	String	Ex.: PPS_Installation/cfg/region_config.cfg
SM_COASTALZONE_LIMIT	If the fraction of land is equal or less than this value, the pixel is classified as open sea and not coast.	Integer	0-255
SM_LANDSEA_FRACTION_MAX	If the fraction of land is equal or greater than this value, the pixel is classified as true land and not coast.	Integer	0-255, recommended value 255.
SM_KERNEL_SIZE	The size of the kernel used for texture calculations in PGE01 and PGE02. Should be matched with SM_THROFFSETS_NAME and	Integer	Ex.: 3 or 5 Do not use other

KEYWORD	Description	Type	Possible Value(s)
	SM_CLOUDTYPE_OFFSETS_NAME, i.e. threshold_offsets.cfg for kernel size 5 and threshold_offsets_gac.cfg for kernel size 3		values!
SM_LINENO_IN_FILENAME	If 1: Products should have scan line number in their file names. If 0: Products use the normal file name convention, without scan line number. The recommended choice!	Integer	0 or 1
PPS_NWP_CONFIG_FILE	The configuration file for reading NWP fields. The file is optional to have, as otherwise default settings will be used.	Integer	${PPS_SOFTWARE}/cfg/pps_nwp.ini

Table 5: PPS environment variables

9. TROUBLESHOOTING

This section gives some solutions to problems that have been encountered by users. Most of them have been answered on NWCSAF Helpdesk.

9.1 IN GENERAL

TBD-03

9.2 QUESTIONS THAT REFERS TO LINUX AND SUN

Q: Which commercial Fortran 90 is appropriate to use on my Linux ?
A: We recommend using *gfortran* rather than any commercially provided fortran compiler.

Q: TBD-03
A: TBD-03

9.3 QUESTIONS THAT REFERS TO PPS PACKAGE

Q: My configuration of ACPG doesn't find hldef.mk, what should --with-hlhdf point to ?
A: If configure of ACPG doesn't find hldef.mk, change configure-parameter to
--with-hlhdf /usr/local

Q: My make install of ACPG loops and has started 300 make-processes, what have I done wrong?
A: Ooops ! This can happen if the cst and testdata-directories are missing.

9.4 QUESTIONS THAT REFERS TO THIRD PARTY SOFTWARE

Q: TBD-03

A: TBD-03

ANNEX A. List of TBC, TBD, Open Points and Comments

TBD/TBC	Section	Resp.	Comment
TBD-01	4.1	SMHI	The descriptions of RAM and CPU usage is valid for an older version of PPS.
TBD-02	5.3	SMHI	The development of binary installation is still undergoing. Binaries for some OS are ready, binaries for some other OS will come afterwhile. There are also some details not decided yet, eg. which versions to use.
TBD-03	6, 9	SMHI	The sections for describing FAQ and common problems are quite rudimentary.
TBD-04	7.1.2	SMHI	The mixing of C and Fortran, for AHAMAP, is not yet described.
TBD-05	5	SMHI	The procedure for handling binary installations as debian packages is not yet fully described.

ANNEX B. Compilers and third party software

10.1 COMPILERS

NB! In order to build the software properly on the supported platforms a native ANSI-C compiler and a native Fortran 90 compatible compiler shall be used. On Linux it is recommended to use the Gnu-C and gfortran compilers, version 4.1.2 or later should be enough. On older (or very old) Linux distributions there may not be any gfortran available or it may be too old. In such cases either try the commercial Intel (version 8 or later) or the Portland Group compilers. Up till today various versions of PPS during the past have been verified on Linux with the Portland Group Fortran-90 compiler, the Intel Fortran compilers (version 7 and 8) or the Gnu Fortran compiler gfortran (with Gnu-C). However, since version 2009 testing has exclusively been done using gfortran. Other (than pgf90 or ifort/ifc) commercial Fortran compilers may be verified to work with PPS in the future. Please observe that the Gnu Fortran 77 compiler is not adequate.

We strongly recommend to use the same C compiler for building the PPS software as was used when building the third party software packages. The same goes for the Fortran compiler.

A list of some available compilers follows, with comments of what can be expected from them. As more compilers are tested, by SMHI or by other users, this information will be updated and provided via NWC SAF Helpdesk.

- Gnu Fortran compiler *gfortran* together with Gnu C *gcc*, versions 4.1.2 or 4.3.2 on Linux Red Hat ES5.1 and Fedora Core 10 respectively: Supported, and verified with PPS-v2010 and v2009.
- Portland Group Fortran 90 Compiler (version 3.2-4) together with Gnu C gcc (versions 3.3.2 or 3.2.3) on Linux: Verified with PPS version 2.0.
- SUN Forte Developer7Enterprise Edition (Fortran 90 and C) on SUN solaris: Verified with PPS version 2.0.
- Intel version 8 and 7 Fortran 90 compiler together with Gnu C gcc (versions 3.3.2 or 3.2.3) on Linux: Not supported, but verified with PPS version 2.0. (Mandrake 10 and Red Hat Enterprise 3). There might be some problems with this compiler for PPS version 2008 –a patch can be made if found necessary.
- Gnu Fortran compiler g77 is not supported and proven to fail with current PPS (See note)

10.1.1 Mixing compilers

In principle it should work to use e.g. the Gnu C-compiler for a third party software package and some commercial (native) C-compiler for the PPS. However, it might at best require extra compiler options or at worst generate erroneous libraries (static or shared). It is a general recommendation to use compilers of the same vendor whenever possible. On Linux, however, as already stated the PPS has only been verified with the Gnu-C and the Portland Group Fortran compilers.

An example of problems mixing C-compilers on a Compaq Tru64: Compiling Proj (see section 10.2.10) with the Gnu C-compiler generates a peculiar error-message during the building of AHAMAP with the native C-compiler. The message reads

Warning: Linking some objects which contain exception information sections and some which do not. This may cause fatal runtime exception handling problems (last obj encountered without exceptions was `libproj.a).`

10.2 THIRD PARTY SOFTWARE

The PPS software depends on and is build with a variety of open source or free-ware applications. In addition it makes use of the AAPP and RTTOV software packages which are both freely available to EUMETSAT member states. Some important details on how it may affect you is given below. The exact release numbers which has been tested, and if any releases that are known to be incompatible with the latest release of the PPS software is given in [RD.3].

10.2.1 The needs for third party software

Not all third party software listed here or in [RD.3] used by PPS is absolutely necessary in all situations. It depends on how the PPS package is run, by running the python modules or only the C-executables, and also depending on what functionality is used. Be aware that the PGE03 for semi-transparent clouds is implemented in python, whereas the PGE03 module for opaque clouds and the other four PGEs have their core parts implemented in C. Though, the python is necessary for the preparation steps, and the python wrap-arounds for the PGEs also have their functionality.

This table describes which software is needed for different modes of operation.

		PY **(Running the PPS modules)**	**C** **(Running the C-executables)**	**Comments**
Gnu Make		X	x	Gnu make is used when compiling the software but it does not need to be available once the software is built and installed
Gnu z-lib		X	x	
HDF5		X	X	PPS is compatible with hdf5-1.8.x but **not** with hdf5-1.6.x or previous
HLHDF		X	x	
SZIP		-	-	Could be required: depending on how HDF5 is installed. If HDF5 was build with szip support it will be needed.
Jpeg lib		X	-	Only needed for image generation.
Proj.4		X	x	

		PY **(Running the PPS modules)**	**C** **(Running the C-executables)**	**Comments**
Python		X	-	
NumPy		X	-	
PIL		X	-	Only needed for image generation.
SciPy		X	-	
h5py		X	-	
netcdf4-python		X	-	
pygrib		X	-	
pyproj		X	-	
pyresample		X	-	Used for remapping VIIRS products to region, and for remapping some GRIB fields in NWP pre-processing.
RTTOV-11		X	x	Only needed for PGE03
GRIB API		x	-	pygrib uses GRIB API for reading GRIB files (edition 1 and 2).
OpenJPEG/ Jasper		(x)	-	Building GRIB API with support for JPEG2000 compression, using either OpenJPEG or JasPer, is recommended.
Autoconf		-	-	It may be needed if the configure script does not work on your system, and a new one has to be generated locally.

10.2.2 Gnu Make

The Gnu Make utility shall be used, rather than any other possible native Make-facility!

10.2.2.1 Affects on the final PPS output

A change in Gnu Make version could result in a failure building AHAMAP and ACPG. But if the package is built successfully a change of version in Gnu Make will not cause any change in the PGE outputs.

10.2.2.2 Known problems and suggested solutions

A common mistake when building the third party and PPS software is to invoke the native *make* instead of Gnu-*make*. It may not always be apparent which *make* is in action. Therefore we recommend that you once and for all define `make` to point at the Gnu-*make*. In tc-shell you may do this in the terminal or add the line to your `.cshrc` file:

NB!

```
> alias make /usr/local/bin/make
```

provided the name of your GNU-*make* is `make` and is placed under `/usr/local/bin`

10.2.3 Gnu z-lib

Gnu z-lib (standard compression library). Needed for HDF5 among other things.

10.2.3.1 Affects on the final PPS output

The zlib library is used by the HDF5 API for the built in compression. PPS uses compression level 6 as default. So a change in zlib version might influence the final output.

10.2.3.2 Known problems and suggested solutions

None.

10.2.4 HDF5

HDF5 – the new generation Hierarchical Data Format, HDF, from the National Center for Supercomputing Applications, NCSA, University of Illinois at Urbana-Champaign, USA.

SZIP may also be needed in your system if HDF5 has been built with SZIP support.

10.2.4.1 Use of SZIP

The default compression algorithm used in HDF5 builds on the zlib compression library. With the latest versions of the hdf5 API library (versions 1.6 and later) it is possible to get support for a supposedly better/more efficient compression (for large image data in particular) method using the szlib library. However, it is not mandatory to have szlib installed or to use this new compression method. One can still go on using the compression provided by zlib which is still adequate and sufficient for PPS.

The Higher Level hdf5 library (hlhdf – see below) developed at SMHI is prepared for the szlib compression but this compression option is not used in PPS. So, szlib and hdf5 can be installed on your machine with support for the new compression method and you can even build hlhdf and pps (acpg) agains it. However, it is not possible to take advantage (if there actually is any great advantage is still an open issue) of the szlib compression in PPS.

As an example we have at SMHI built hdf5 like this:

```
./configure –-prefix=/local_disk/opt/HDF5/1_8_4
```

(It might be a good idea not to elaborate too much adding many options when building the hdf5 library. We have noticed that some options may cause errors when running *make check* after having run *configure*.)

And HLHDF is built like this:

```
./configure --prefix=/local_disk/opt/HLHDF --with-
hdf5=/local_disk/opt/HDF5/1_8_4/include,/local_disk/opt/HDF5/1_8_4/lib --with-
numpy=/usr/lib64/python2.5/site-packages/numpy/core/include/numpy
```

10.2.4.2 Affects on the final PPS output

ALL PGEs and intermediate/temporary results are stored using the HDF5 file format. However, a minor change in HDF5 version is not likely to cause a change in the actual output affecting the PGE results. It might be that PPS (and HLHDF) is not compatible with a new version of HDF5 but change in version should not normally affect the data content. This would rather be the case if there is a change in the compression method or library used.

The actual SZIP compression algorithm is not used in PPS at the moment. A change in version will have no effect on the PPS output.

10.2.4.3 Known problems and suggested solutions

Currently we have no known problems. But please observe that it is required to use hdf5-1.8.x or later. The PPS-v2010 and onwards have left the old 1.6 branch, though both branches (hdf5 1.6.x and 1.8.x) are being maintained by the HDF Group. HDF5 files written with PPS-v2010 and hdf5-1.8.x may be readable also with hdf5 1.6.x. Our very basic tests showed that h5dump is able to correctly read the files with hdf5 versions down to 1.6.3. The PPS files written with hdf5-1.8.x may also be readable using hdf5 version 1.6.2 and earlier, but we have done no tests with those versions.

10.2.5 netCDF4

NetCDF (network Common Data Form) is a set of software libraries and machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data.

10.2.6 HL-HDF

HL-HDF – the Higher Level interface to the HDF5 file format, developed at and available from SMHI. HL-HDF is today (and since a few years back) compatible with NumPy (and not Numeric), and also requires hdf5-1.8.x

NB! As part of the PPS software needs the HL-HDF python interface Pyhl, remember to have it installed somewhere along your PYTHONPATH. The standard way is to have `_pyhlmodule.so` placed under the <python-lib>`/site-packages/Pyhl` directory and having a Pyhl.pth file with the string Pyhl in it (the relative path to where the package is placed). This is just as you probably have done with PIL, Numeric, etc.

Refer to [RD.3] for compatibility issues.

10.2.6.1 Affects on the final PPS output

See 10.2.4.2

10.2.6.2 Known problems and suggested solutions

None.

10.2.7 jpeg-library

Standard library for JPEG compression needed by PIL. If it is not installed the PPS should still be able to run, but the output image format chosen should be something else than JPEG (TIFF or PNG for example).

10.2.7.1 Affects on the final PPS output

The jpeg-library is used inside PIL and will affect the image outputs only if using the jpeg image format.

10.2.7.2 Known problems and suggested solutions

None.

10.2.8 Proj.4

Proj.4 – The USGS Projection library version 4 (a Unix standard tool for working with map-projections) is needed.

10.2.8.1 Affects on the final PPS output

The USGS Proj.4 library is used extensively in the PPS package for mapping data. Thus a change in Proj.4 version could result in changes of the PGE outputs.

10.2.8.2 Known problems and suggested solutions

None. PPS version 0.3.0 and later are compatible with both the latest and previous versions of Proj.4. See Software Versions Document ([RD.3]) for more details.

10.2.9 Python

A Python installation shall be available.

10.2.9.1 Affects on the final PPS output

Python is used everywhere except in the core PGE code for PGE01, PGE02, PGE04 and PGE05. A change in Python version could in principle cause changes in the PGE-03 output or in the PGE product images derived from the hdf5 output.

10.2.9.2 Known problems and suggested solutions

PPS has no yet been adapted to python-3.

If some python packages have to be installed at other places than the recommended paths it might be needed to modify the PYTHONPATH and/or the .pth configuration files for the python interpreter. It is easiest to explain how these two options work by writing an example.

Let's say that there is a python module called mymodule.py located in the directory /home/user1/pythonmodules and that it should be possible to import this module by doing

```
%>>> import mymodule
```

- PYTHONPATH
  This is an environment variable that is behaving similar to the PATH environment variable, that is a list of paths separated by a colon ':'.
  ```
  %> setenv PYTHONPATH /home/user1/pythonmodules
  %> python
  %>>> import mymodule
  ```

- <file.pth>
  When the python interpreter is started, it looks in a catalogue called site-packages for all .pth files and adds the content of each .pth to the search list.
  ```
  %> echo "/home/user1/pythonmodules" > xxx/lib/python2.5/site-packages/mymodule.pth
  %> python
  %>>> import mymodule
  ```

10.2.10 Third party Python packages

The following Python modules/packages, not in the Python standard library, are needed:

- NumPy library. Used for PGE03 module for semi-transparent clouds, and for generation of summary statistics on the PGE outputs. Also used throughout the PPS system in general when operating on the data in Python. Often automatically available on a Linux distribution.
- PIL (the Python Image Library from Secret Labs AB). Often automatically available on a Linux distribution
- sciPy
- h5py
- pygrib
- pyproj
- pyresample
- netcf4-python

10.2.10.1 Affects on the final PPS output

NumPy is used extensively in PPS. However, it is not used in the actual core PGE code except PGE03. A change here could in principle cause changes in the PGE outputs.

PIL is used for generating images of the outputs or intermediate results. A change in PIL will not have any effect on the hdf5 outputs.

The Tk/Tcl and Tkinter libraries are used for the Task Manager GUI (and PPS Viewer) and a change in version will not affect the final PGE outputs.

The Python Mega Widgets library is used for the Task Manager GUI (and PPS Viewer) and a change in version will have no effect on the final PGE outputs.

SciPy is required by PGE03 and PGE04 pre-processing, and NWP extraction. A version change could affect PGE outputs.

H5py is required for some HDF5 operations in PGE04 pre-processing, PGE05 processing and VIIRS preprocessing. A version change is unlikely to affect PGE output products.

Pygrib and pyproj are required for NWP pre-processing. A version change could affect PGE outputs

pyresample is required to remap VIIRS products to region, and to remap GRIB fields not in regular or rotated lon/lat or gaussian grid projection (e.g. OSISAF sea ice map). A version change could affect PGE outputs.

netcdf-python is used for converting the PPS output files to netCDF, and for reading OSISAF sea ice map, if they are in netCDF.

10.2.10.2 Known problems and suggested solutions

Versions 1.9.0 and 1.9.1 of pygrib are incompatible with ECMWF global lat/lon GRIB projections. Version 1.9.2 is recommended.

See Software Versions Document ([RD.3]).

10.2.11 argparse

From python 2.7 argparse is included in the Python standard library.

For users of python, with versions earlier than 2.7, argparse is a separate package.

10.2.12 RTTOV

RTTOV – A Fast Radiative Transfer Model (RTM) maintained and distributed by the NWP-SAF.

The pre-processing module for PGE03, preparing the data for the online processing of the CTTH product, make calls to functions/subroutines provided by RTTOV (via the static RTTOV library). PPS currently builds on RTTOV-11.

10.2.12.1 Affects on the final PPS output

A change in RTTOV version will have effects only on the results of PGE03. This is when nothing else is changed: The threshold tables for PGE01&02 are generated off line with RTTOV and these remains constant for each version of PPS and are only updated occasionally between PPS versions.

10.2.12.2 Known problems and suggested solutions

Please observe that the PPS version 2014 is adapted for and tested with RTTOV-11. There is no support for other versions of RTTOV. See Software Versions Document ([RD.3]).

So far, there are no known problems in combining PPS with the recent RTTOV-11 package. For installing RTTOV-11, simply uncompress and unpack the tar-ball. Then follow the instructions in the 'readme.txt' file. If you are going to build a 'slim' version, make sure that the installation consist at least of the following directories: 'bin', 'include', 'lib', 'mod' and the 'rtcoef_*' directories. Note, that the version-number in the rtcoef file-name marks just the compatibility-level. So even if you run RTTOV-11, the 'rtcoef_rttov7' directory is needed.

10.2.13 GRIB API

GRIB API for reading NWP Grib data (available from the ECMWF). This library is used by the Python module pygrib, and must be installed prior to building pygrib.

10.2.13.1 Use of OpenJPEG or JasPer

ECMWF recommends that GRIB API is built with support for JPEG2000 compression, through either the OpenJPEG or JasPer library, for a standard installation. It is also possible to build GRIB API without support for JPEG2000, by using the --disable-jpeg configure option.

There is a link to jasper from the GRIB API site. To work with pygrib, JasPer needs to be built with shared libraries (libjasper.so). This is not done by default, but can be accomplished by adding --enable-shared to the JasPer configure command. (The JasPer packages included in the RedHat Enterprise Linux distribution include shared libraries.)

10.2.13.2 Affects on the final PPS output

The GRIB API is used by pygrib to read GRIB data, and changing the version of this software might cause changes in the content of the remapped NWP parameters (the files under `import/NWP_data/remapped` named "S_NWC_nwp_<parameters accronym>_*.h5" or the files under `import/ANC_data/remapped` named "S_NWC_ctth_segment_*.h5") and will then have effects on all PGE outputs.

10.2.13.3 Known problems and suggested solutions

GRIB API needs to be compiled with `CFLAGS=-fPIC`, to be usable in pygrib.

10.2.14 AAPP

Since PPS version 2014 the AAPP navigation library is not used anymore.

However, AAPP is needed in order to run PPS with global Metop GDS granules as input. PPS makes use of a format conversion script that converts the EPS level 1 data to AAPP level1b format. This usage requires that AAPP is available, but it does not require that PPS is compiled with AAPP.

Also AAPP is of course needed to produce the level 1b/1c files which are input to the PPS-package, using direct readout NOAA and Metop data. PPS provides no support using level 1b/1c data not produced by AAPP.

10.2.14.1 Affects on the final PPS output

The AAPP package is only needed in PPS when generating Global Metop products.

Both the ACPG and the AHAMAP sub-packages of PPS should be built without AAPP support. Use the autconf directive `–without-aapp`!

10.2.14.2 Known problems and suggested solutions

Earlier AAPP versions (prior to version 4.4) did not compile straight away using a fortran 90 compiler. However, the PPS/ACPG requires fortran 90 as the RTTOV requires fortran 90 compilation. This has made it difficult to streamline the PPS built process, and at least on Linux using the Portland Group fortran compiler suite requires the manual intervention of one makefile in ACPG. See section 6.4.

10.2.15 ANA – The Automatic Navigation Adjustment Technique

The NOAA and Metop satellite data are geolocated or navigated using the navigation library available in AAPP and orbital parameters (bulletins) provided as input to AAPP. AAPP can take several types of orbital bulletins. At SMHI we use Two Line Element (TLE) files provided from *Space Track* on the internet. Also TBUS and SPM files are supported – see AAPP documentation for further details.

However, even with fresh and up to date TLE files or equivalent, the AVHRR navigation is never perfect on the NOAA satellites. The Metop spacecrafts are navigated more accurately, however, using a GPS instrument on-board the spacecraft. On the NOAA satellites often large navigation errors of several km can be observed, if no attempt to automatically adjust for the navigation errors.

The position of an AVHRR footprint depends on time, satellite position and velocity, satellite attitude (its orientation) and radiometer viewing geometry. The radiometer geometry is known prior to launch. Time is usually available, e.g. through the satellite time corrected from the satellite clock error, or from an independent clock. Satellite position and velocity may be calculated by an orbit prediction model ingesting daily bulletins (e.g. TLE). The remaining unknown is the satellite attitude, or in fact how the actual attitude deviates from its nominal value.

The actual satellite attitude can be estimated if an adjustment is performed on the raw data (navigated using the nominal attitude) using known landmarks, as has been done operationally at CMS, Météo-France since 1990, using their Automatic Navigation Adjustment (ANA) technique (Brunel and Marsouin, 2000). ANA has been used in operation at SMHI since 2003.

ANA combines a physical image deformation model and automatic adjustment on coastal landmarks. The navigation adjustment is done in satellite co-ordinates allowing interpreting the landmark navigation errors in terms of satellite attitude: yaw, pitch and roll.

More details on ANA can be found in [RD.15] and [RD.16].

NB! ANA is **not required** to build PPS, nor is it required for running PPS. But if you like to have improved data quality on input you should consider installing it and running it in concert with AAPP.

10.2.16 AutoConf

The configure scripts, when attempting to configure the build environment, has been created by using `autoconf` which is a macro language for simplifying this task. The user shall not need to run the `autoconf`, as the configure script is already generated at SMHI and included in the software package. Thus, normally the user does not need to have the AutoConf software installed.

However, a macro conflict could appear while the macros changes between operating systems – as has been seen e.g. for Fedora Linux. Then Autoconf will probably be needed, after adequate update to the configure.ac files in PPS.

10.2.16.1 Known problems and suggested solutions

There are some known limitations in this macro language related to the decisions of fortran compiler. Autoconf does not differentiate between F77 and F90 compilers, this is why there is a F77 variable in the Makefiles even though we prefer to use F90. The configure script will, however, always try to locate a F90 compiler before it tries to locate a F77 compiler unless the environment variable F77 has been set. In case the F77 environment has been set the searching is not performed and the compiler specified in the F77 environment variable will be used.

If, on a Linux system, for instance, and there is no access to a commercial fortran compiler and the F77 environment variable is not set, the g77 will be found. The g77 is, however, not supported and not adequate for compiling PPS and its required 3rd party software. Instead we urge the user to try to use the newer g90/g95 compiler, which has proven to work with PPS-v2008 on Ubuntu. Try set the F77 variable to the name of your fortran compiler and try to build the 3rd party software and PPS. NB: If F77 is not set, the acpg and ahamap configure scripts try to find a suitable fortran compiler for you, from a predefined list of compiler names, gfortran is included since version 2008.