AAPP v8 Installation Guide | NWP SAF

ATOVS and AVHRR Pre-processing Package (AAPP)

Document: NWPSAF-MO-UD-005
Version: 8.2
Date: July 2020

Author: Nigel Atkinson (Met Office)

This documentation was developed within the context of the EUMETSAT Satellite Application Facility on Numerical Weather Prediction (NWP SAF), under the Cooperation Agreement dated 29 June 2011 between EUMETSAT and the Met Office, UK, by one or more partners within the NWP SAF. The partners in the NWP SAF are the Met Office, ECMWF, DWD and Météo France.

Change record			
Version	Date	Author / changed by	Remarks
8.0	Dec 2017	Nigel Atkinson	This document previously existed as 6 separate html files. For this release it has been brought into one document and extensively edited.
8.1	Apr 2019	Nigel Atkinson	Update links
8.2	July 2020	Nigel Atkinson	Add details about building ecCodes with jpeg2000 support

Introduction

This document describes the installation of the ATOVS and AVHRR Processing Package (AAPP), which is a deliverable of EUMETSAT's Numerical Weather Prediction Satellite Application Facility (NWP SAF). For details on the capabilities of AAPP, please see the companion document the AAPP User Guide.

The currently available software has been developed for UNIX and Linux systems. Users who want to adapt the software to other operating system environments are obliged to do this without installation support. AAPP has been tested by the NWP SAF on Linux and macOS operating systems. It has also been run on AIX systems, though currently only limited support can be provided for this type of system.

The following is required for the installation of the main AAPP modules:

- UNIX or Linux operating system (32-bit or 64-bit)
- Korn shell
- A Fortran90 Compiler
- A C compiler
- make utilities
- perl5, lex, yacc
- gunzip and tar
- about 9GB of disk space

External libraries are required to run some parts of AAPP (detailed more fully in a later section):

- A BUFR library, either BUFRDC or ecCodes, both available from ECMWF
- HDF5 libraries (available from The HDF Group)
- SZIP and ZLIB compression (for use with HDF5 – see http://www.hdfgroup.org/HDF5/release/obtain5.html#extlibs)

The IASI local processing software, OPS-LRS, is delivered separately from the main AAPP, and its installation is described in the OPS-LRS User Manual.

Delivered items

AAPP is normally delivered via the NWP SAF web site. Each user needs to first register on the web site. Then select the "software downloads" button and follow the instructions to select the required deliverables (e.g. AAPP and OPS-LRS). A link to the latest full release will then be displayed, which can be downloaded as a gzipped tar file. This file contains source code and the majority of the required data files. You may also need to download some of the update packages, which allow building of a specific release, if required.

Some of the larger data files must be downloaded separately. These can be downloaded via https://nwp-saf.eumetsat.int/site/software/aapp/download/. If you plan to run the MAIA cloud mask then you will need the MAIA4 data file, and if you plan to run OPS-LRS (the IASI level 1 processor) you will need the OPS-LRS auxiliary data. Note that a script is available to simplify the process of downloading the auxiliary data. Test cases are also available for download.

An important part of each release is the Release Note. This will give details on the changes made and will provide guidance on how to build or update your installation. Release notes can be found on the Release History page.

You may also need to consult the other documents on the AAPP documentation page. For example, the Scientific Description, the Software Description and the Data Formats documents.

Installation

The procedure to install AAPP can be summarised as follows:

1. Download the required AAPP files to your workstation and unpack them
2. Install any external libraries that you may need
3. Configure the installation to suit your workstation
4. Build AAPP by runing "make"

The paragraphs below describe the various steps required to build AAPP, but you may also wish to consult the example build scripts that are available in the downloads directory, in particular install_aapp8.sh. Save the script to your workstation and run *chmod +x install_aapp8.sh*. You can run the different parts of the script in turn, first to download and install specified versions of the external libraries, then to configure and install AAPP and OPS-LRS, and finally to download the necessary data files. You can customise the build script to suit your requirements.

Download and unpack AAPP

We assume the use of AAPP v8.1 in the example below, but it could be any full release.

Copy the AAPP_8.1.tgz file and (optionally) the AAPP_MAIA4_thresholds_v8.tgz and AAPP_MAIA4_atlas.tgz files to a suitable location on your system and unpack them like this:

```
tar -xzf AAPP_8.1.tgz
cd AAPP_8.1
tar -xzf ../AAPP_MAIA4_atlas.tgz
tar -xzf ../AAPP_MAIA4_thresholds_v8.tgz
```

If you wish, you can re-name the AAPP_8.1 directory after the first step. It is referred to in this document as the AAPP top directory.

Some older versions of tar do not support gunzip, in which case you can do:

```
gunzip -c AAPP_8.1.tgz | tar -xf -
```

If you need to install an update release (having previously installed a full release), go to the AAPP top directory and type

```
tar -xmzf AAPP_update_8.x.tgz
```

where x is the update number and the -m flag ensures that the newly-unpacked files have a current time stamp and will therefore be picked up by the subsequent "make".

Install external libraries

The table below lists the external libraries that are most likely to be needed, their use in AAPP, and how to build them (after unpacking). In the table, yyy and zzz indicate directories that you have to specify on your system. Some of these libraries (e.g. HDF5) can be downloaded as binaries, but the table assumes you are building from source. Most users will not need to install all the libraries, just the ones needed for your application.

Library	Uses in AAPP	Available from	Install scripts	Notes
BUFRDC	Legacy BUFR library. Used in AAPP for encoding and decoding of BUFR datasets from a range microwave and infrared sounders.	ECMWF	./build_library ./install	Fortran77 and C code. The build_library interactive script offers you a choice of compilers and installation location. Use the same compiler for AAPP. Environment variable BUFR_TABLES is needed at run time. EMOSLIB may be used as an alternative to BUFRDC.
ecCodes	BUFR and GRIB library from ECMWF. Has largely superseded GRIB_API and will eventually replace BUFRDC. For BUFR encode/decode, the AAPP v8 user can choose whether to use the BUFRDC interface or the ecCodes interface. For MAIA, GRIB forecast files are used as input. It is recommended to use ecCodes for this, but GRIB_API will work as an alternative. The choice should be made at the AAPP configure step; you should not build AAPP with both ecCodes and GRIB_API. Note: ecCodes is relatively new, so if you encounter a problem, please report it to software.support@ecmwf.int	ECMWF	If "cmake" is not already installed on your system, you will need to install this first. cd $build_dir cmake \ -DCMAKE_INSTALL_PREFIX=${install_dir} \ -DCMAKE_Fortran_COMPILER=${FC} \ $source_dir make make install >install.out	The environment variables and directories could be set up as follows (using v2.5.0 as an example; it is recommended to use the latest release): FC=gfortran source_dir=$PWD/eccodes-2.5.0-Source build_dir=$PWD/build_2.5.0_${FC} install_dir=$PWD/install_2.5.0_${FC} rm -rf $build_dir $install_dir mkdir -p $build_dir $install_dir On some systems you may need to build a static version of the library, using the flag -DBUILD_SHARED_LIBS=OFF If you will be working with GFS forecast files (for MAIA4) then it is important to ensure that ecCodes is built with jpeg2000 support – via the OpenJPEG or JasPer libraries. Normally cmake will detect installed copies of these libraries when building ecCodes, e.g. /usr/lib64/libjasper.so. However, if you have them installed in a non-standard location you can use the cmake directive -DCMAKE_PREFIX_PATH=[directory]. If jpeg2000 is correctly enabled, you should see the text "HAVE_JPEG=1" in ecbuild.log, after the "cmake" step.

HDF5	1) Reading Sensor Data Record files for ATMS, CrIS, VIIRS, MWHS, MWTS, IRAS.
	2) Reading IASI eigenvector files.
	3) Creating the MAIA output files.
	4) Generating .h5 versions of the various .l1c datasets.
		The HDF group	#For AIX:
			FCFLAGS="-qextname"
			export FCFLAGS
			
			#For all systems:
			./configure [options]
			
			make
			make install
				Typical configure:>
				
				./configure --prefix=yyy \
				--enable-fortran
				
				The Fortran interface (libhdf5hl_fortran) is required if you wish to use MAIA. It is not required for the other applications.
				
				You may also need (see below):
				
				--with-szlib=PATH_TO_SZIP
				
				For MAIA, it is best to use HDF5 version 1.8.8 or later (a patch is available if you want to use earlier versions – contact the NWP SAF Helpdesk for details).

				Note for AIX xlf: By default, AAPP is built with the "-qextname" compiler flag; in which case you should use it for the hdf5 library also.
SZIP and ZLIB	May be used by HDF5.	The HDF group	./configure −−prefix=zzz make make install	These libraries are not explicitly used by AAPP, but they may be needed for your installation.
GRIB_API	Although GRIB_API is still available from ECMWF, it has now been superseded by ecCodes, so the description has been removed for this version of the document.			
Flex	Needed for OPS-LRS. Often comes with Linux distribution, but not always.	Github. The latest release should work, or use 2.5.35.	./autogen.sh ./configure −−prefix=zzz make make install	For OPS-LRS use, you can copy the libfl.a to OPS-LRS directory src/EXT /env/lib
gettext	autopoint is needed to install flex. Often comes with Linux distribution, but not always.	ftp.gnu.org	./configure −−prefix=zzz make make install	
jasper	jpeg2000 support when reading GFS forecast files in MAIA4	Univ of Victoria	mkdir -p $build && cd $build cmake -G "Unix Makefiles" -H$src -B$build \ -DCMAKE_INSTALL_PREFIX=$install \ -DCMAKE_BUILD_TYPE=Release \ -DCMAKE_SKIP_INSTALL_RPATH=YES \ -DJAS_ENABLE_DOC=NO make make install	

Note that some other external libraries (e.g. xerces) are also needed to build OPS-LRS. These are available from the AAPP Download page. See the OPS-LRS User Manual for details.

As a guide, the requirements for external libraries, depending on your application, are listed below:

- Processing raw ATOVS and AVHRR data to level 1c or 1d: **none**
- Processing IASI using OPS-LRS: **xerces** and **fftw**
- Converting to hdf5, or reading IASI eigenvector files: **HDF5 C interface**
- BUFR encode / decode (e.g. DBNet data handling): **BUFRDC** and/or **ecCodes**
- Running MAIA: **HDF5 Fortran interface** and either **ecCodes** or **GRIB_API**

Configure AAPP

This section is split into 2 parts: firstly the instructions for configuring a "core" AAPP installation with no external libraries (which may be adequate for some users), and secondly the instructions for creating a fuller installation. Note that the procedure uses a perl script named configure that is located in the AAPP top directory. (It does not use GNU autotools).

Configure the "core" AAPP

All that is required to install "core" AAPP is to go to the AAPP top directory and run the configure script with your site-id specified as:
−−site-id= a 3-character abbreviation of your centre
and the Fortran compiler you are using as
−−fortran-compiler= any of : gfortran ifort pgf90 g95 AIX AIX-Hitachi
For example, to configure AAPP for a Linux system using gfortran you could type (from the top directory)
./configure −−site-id=UKM −−fortran-compiler=gfortran
There are additional options (type ./configure from the top directory to view them) that you may wish to specify, such as your local station name, and some must be specified to obtain full AAPP functionality as described below. But to complete the "core" AAPP installation type
make

Check that the installation was successful and that you did not see any errors. If it was OK you may start testing AAPP. You will be able to run the NOAA test case and parts of the Metop test case.

Configure your "fuller" AAPP installation

Go to the AAPP top directory and type ./configure to see the options.

You have to specify the following:

−−site-id= a 3-character abbreviation of your centre
−−fortran-compiler=any of : gfortran ifort pgf90 AIX AIX-Hitachi

Note 1: To see the complete list of compilers that have been used in past versions of AAPP, check the contents of the files in directory config. However, some of these are no longer supportable by the NWP SAF due to their age and lack of test platforms. In particular, we no longer recommend the use of a FORTRAN77 compiler such as g77, as many parts of AAPP will not build. Also g95 does not work with the MAIA4 calls in avh2hirs.

Note 2: Some Fortran 90 compilers require different flags to indicate the directory to search for Module files. The default is "-I", but you may need "-M". If necessary, you can adjust the applicable file in the config directory (and please inform the NWP SAF Helpdesk if you find something that needs changing).

In addition, you can optionally specify:

−−prefix= installation prefix, if you wish to install the AAPP executables and data files to somewhere other than the current directory e.g. $HOME/AAPP_8_runtime
Note: do *not* choose this directory to be the same as the AAPP top directory, otherwise all your source files will be deleted!
−−station= your local station name (see list in AAPP/src/navigation/libnavtool/stations.txt). The default is "dummy_station"
−−station-id= your local station ID, for "datset name" field in the level 1a/1b header. The default is WE (Western Europe)
−−external-libs= location and options for external libraries (in quotes), e.g. "-L${DIR_ECCODES}/lib -leccodes -leccodes_f90

-L${DIR_HDF5}/lib -lhdf5 -lhdf5_hl -lhdf5_fortran -lhdf5hl_fortran".
––external-includes= location and library options for external include files, e.g. -I${DIR_HDF5}/include
––external-modules= location for external Fortran90 modules, e.g. -M ${DIR_ECCODES}/include
Note: some compilers use "-I" instead of "-M".
––c-compiler= name of your C compiler (for use with METOP tools, IASI tools and HDF5).
––tle-user= www.space-track.org username; required for using tle retrieval from the internet.
––tle-passwd= www.space-track.org password
––scriptdir= optional directory in which to place the AAPP shell scripts. Normally the scripts and .exe files are placed together in "bin" directories, but on some systems, notably Cygwin, they must be separated, e.g. specify "scripts".

If an external library is centrally installed on your system (e.g. in /usr/lib), you will not need to specify -L under ––external libs, but you will need to link to it using -l. Similarly, if your external library has centrally-installed include files (e.g. in /usr/include) then you will not need to specify ––external-includes.

Regarding HDF5: if you do not need to run MAIA, and just want the HDF5 conversion tools, it is sufficient to just specify "-lhdf5". Some systems may require -lz.

You will probably need to use the same Fortran compiler for AAPP as was used to build the HDF5 Fortran interface. You may also need some of the same compiler flags (e.g. "-qextname" for the AIX xlf compiler).

If the Fortran compiler you wish to use is not listed, you may create your own configuration file in the *config* directory, taking one of the existing ones as a template. If you do so successfully, please inform the NWP SAF, for the benefit of other users.

If your station is not contained in the station list file, AAPP/src/navigation/libnavtool/**stations.txt**, you can edit and update the file if necessary before you go on to build AAPP. Please notify the NWP SAF of the coordinates of your station, so that it can be included in a future update.

The "station", "station_id", "tle_user" and "tle_password" options are optional. The values that you enter are stored in the ATOVS_ENV8 file. You may edit this file later if you want to change the values.

When you build AAPP, the source, data files and executables are initially placed in a fixed directory tree structure as follows:

AAPP_8/AAPP/bin
AAPP_8.1/AAPP/data
AAPP_8.1/AAPP/src
AAPP_8.1/metop-tools/bin
AAPP_8.1/metop-tools/src
etc.

This will be fine for many users. However, if the "prefix" option is specified then you can install elsewhere the *bin* and *data* directories, and other files needed at run time. This is done via the "make install" command. The path that you enter is also stored in the ATOVS_ENV8 file, as environment variable AAPP_PREFIX. If you do not use the "prefix" option then AAPP_PREFIX is set to the AAPP top directory.

Thus the run-time directories are always $AAPP_PREFIX/AAPP/bin, $AAPP_PREFIX/AAPP/data, etc.

For example, an installation with hdf5, BUFRDC and ecCodes capability could be set up like this:

```
hdf=/path_to_hdf5
bufr=/path_to_bufrdc
ec=/path_to_eccodes

[[ ${LD_LIBRARY_PATH:-} = *${hdf}* ]] || export LD_LIBRARY_PATH=${LD_LIBRARY_PATH:+$LD_LIBRARY_PATH:}${hdf}/lib
[[ ${LD_LIBRARY_PATH:-} = *${ec}* ]] || export LD_LIBRARY_PATH=${LD_LIBRARY_PATH:+$LD_LIBRARY_PATH:}${ec}/lib
configure --station=exeter --fortran-compiler=ifort --site-id=UKM \
    --external-libs="-L$bufr/lib -lbufr -L$ec/lib -leccodes -leccodes_f90 -L$hdf/lib -lhdf5 -lhdf5_hl -lhdf5_fortran -lhdf5hl_fortran" \
    --external-includes="-I$hdf/include -I$ec/include"
```

If a library is centrally installed on your system you will not need to modify LD_LIBRARY_PATH. The error messages from the build step should tell you if any libraries are missing.

The *configure* script creates a file Makefile.ARCH in the top directory, which contains the system settings and is referenced by all the lower-level makefiles. You may wish to modify the contents of this file before proceeding, for example:

- to change some of the compiler flags. In particular you are advised to check that the optimisation flags are suitable for your system.
- to provide an explicit path to the Fortran compiler.

Some components of AAPP may be customised by the user. For example, the IASI level 1d format is defined by the file *AAPP/include/iasi1d.h*. This file has parameters that define the maximum number of Principal Components scores that can be accommodated in the 1d file, and the maximum number of channels. If you want something other than the default (500 channels, plus 300 scores) then you should modify this file before building the software. The number of channels or scores actually *used* may be set at run time, provided it is less than the maximum.

Build AAPP

To build the complete package you just need to do one of the following:

If you did not specify an installation prefix in the configure step, type

make

If you did specify an installation prefix, type

make install

There are a number of other possibilities that users requiring flexibility may adopt:

- AAPP v8 comprises three main components: (i) the core AAPP, (ii) METOP tools and (iii) IASI tools. To build or re-build just one component, cd to the appropriate directory (AAPP, metop-tools or iasi-tools) then run *make* as before. Note that metop-tools must be built before iasi-tools. However, it is advisable to first run *make* from the top of the AAPP tree in order to avoid problems with library dependencies.
- To clean the whole of AAPP, i.e. to delete all the binaries, libraries, data files and object files in the AAPP_8 tree (not the installation tree), type

make clean

- To clean part of AAPP (e.g. AAPP, metop-tools, iasi-tools or a lower-level module), go to the appropriate directory and type

make clean

- To clean the object files of AAPP (or metop-tools, or iasi-tools), leaving the data files and executables unchanged (this can be useful on operational systems), go to the AAPP, metop-tools or iasi-tools directory and type

make clean_obj

- To re-build a lower-level module of AAPP (e.g. amsubcl), cd to the appropriate library (e.g. AAPP/src/calibration/libamsubcl) and type

make lib
cd ../bin
make bin

- To re-install selected data files, cd to the sub-directory of src that contains the data file and type

make dat

- AAPP is delivered with Makefiles already supplied. To re-generate all makefiles (e.g. if you have added a module), go to the top directory and type

perl makefile.PL

If *make* fails because an external library is missing, locate the library and modify your Makefile.ARCH or re-run configure. Remember that -L specifies the location of the library (if it is not in one of the standard directories) and -lx links the library named libx.a. For example, on the ECMWF AIX system, when building AAPP with HDF5 support it was necessary to add -lz to link the centrally-installed zlib.
If the make fails for reasons that are not obvious, before re-trying it is recommended that you remove any partially installed components using make clean. Similarly if having built AAPP you decide that you want to install the BUFR or GRIB libraries, then you can make clean, re-run configure, then re-run make.

The ATOVS_ENV8 file

In AAPP v1 to v5 the ATOVS_ENV file that sets up the AAPP environment was always located in the user's $HOME directory. This caused problems for some users and made it difficult to run different versions of AAPP simultaneously. For AAPP v6 the system was changed, and it is the same in AAPP v7 and v8:

- The script that sets up the environment is called ATOVS_ENV8.
- ATOVS_ENV8 is created in the installation directory by the configure script.
- Also in the AAPP installation directory is a file called ATOVS_CONF. This file is run by all the built-in AAPP scripts.
- When ATOVS_CONF is called it first looks for an ATOVS_ENV8 file in $HOME. If one exists, it will be used.
- If there is no $HOME/ATOVS_ENV8 then the ATOVS_ENV8 file in the AAPP installation directory will be used.

The user may if required move the ATOVS_ENV8 file to his $HOME directory (or create a link). Then to set up the environment for AAPP the user's script (or interactive shell) needs:

. $HOME/ATOVS_ENV8

Alternatively the user's script may source ATOVS_CONF:

AAPP_PREFIX=.....
. $AAPP_PREFIX/ATOVS_CONF

In either case all the AAPP scripts and binaries (including those in metop-tools and iasi-tools) will then be accessible by name, since ATOVS_ENV8 sets up $PATH to include the necessary directories.

You may need to edit the ATOVS_ENV8 file after it has been created by configure – e.g. to set the correct path for the GMT executables (rarely used these days) and to set $LD_LIBRARY_PATH for use with HDF5 or ecCodes. Note that older versions of the HDF5 library (5-1.8.1 and earlier) required the SZIP library to be centrally installed or included in $LD_LIBRARY_PATH.

Special considerations for Windows PC installation

Previous versions of the AAPP Installation Guide gave hints on running AAPP under (i) Microsoft Windows Services for Unix (SFU) and (ii) Microsoft Subsystem for UNIX-based Applications (SUA). However, these are considered obsolete so those sections have been removed. It may still be possible to run AAPP under Cygwin, but this has not been tested by the NWP SAF. The notes made some years ago are included below.

Cygwin is available from http://www.cygwin.com/.As well as standard utilities, you need to install perl, a korn shell (pdksh.exe) and a suitable Fortran compiler. The following changes are needed:

- Ensure the LIB environment variable (echo $LIB) does not contain any Windows path names with colons or spaces, otherwise make will fail. You can nullify this variable by typing:

LIB=

- The AAPP scripts require that the korn shell executable exists as /bin/ksh. So create a link as follows

ln -s /bin/pdksh.exe /bin/ksh

- You should use the −−scriptdir=scripts option in configure. If you do not then the scripts will not be installed; this is because Cygwin will not create prog if prog.exe already exists in the same directory.

- Having run configure, ensure that your PATH definition in ATOVS_ENV8 does not contain any Windows path names with spaces. Edit ATOVS_ENV8 if necessary.

- On some systems, make fails because it does not recognise "\cp" in the Makefiles. If this is the case, edit Makefile.PL, change "\cp" to " cp", change "\rm" to " rm" and "\mkdir" to " mkdir" (i.e. space instead of \). Then type

perl Makefile.PL

AAPP directory structure

After the successful installation of AAPP on your machine you should find on your system the following AAPP directory structure:
./AAPP
./AAPP/bin
./AAPP/gmt
./AAPP/include
./AAPP/lib
./AAPP/man
./AAPP/man/man1
./AAPP/man/man3
./AAPP/man/man5
./AAPP/module

```
./AAPP/orbelems
./AAPP/orbelems/ephe
./AAPP/orbelems/orb_elem
./AAPP/orbelems/satpos
./AAPP/orbelems/tbus_db
./AAPP/orbelems/tbus_db/2010-03
./AAPP/orbelems/tracking
./AAPP/src
./AAPP/src/calibration
./AAPP/src/calibration/bin
./AAPP/src/calibration/libamsuacl
./AAPP/src/calibration/libamsubcl
./AAPP/src/calibration/libavhrcl
./AAPP/src/calibration/libcal
./AAPP/src/calibration/libhirscl
./AAPP/src/calibration/libhirscl_algoV4
./AAPP/src/calibration/libmhscl
./AAPP/src/calibration/libmsucl
./AAPP/src/decommutation
./AAPP/src/decommutation/bin
./AAPP/src/decommutation/libdecom
./AAPP/src/libauxaapp
./AAPP/src/libbufrdummy
./AAPP/src/maia4
./AAPP/src/maia4/bin
./AAPP/src/maia4/libaapp_viirs
./AAPP/src/maia4/libavhrrin_tools
./AAPP/src/maia4/libmaia4
./AAPP/src/maia4/libmaia4IO
./AAPP/src/maia4/libmapviirscris
./AAPP/src/navigation
./AAPP/src/navigation/bin
./AAPP/src/navigation/libMSLIB77_V3.1
./AAPP/src/navigation/libbrolyd
./AAPP/src/navigation/libephtrack
./AAPP/src/navigation/libnavnoaa
./AAPP/src/navigation/libnavtool
./AAPP/src/navigation/libsgp
./AAPP/src/navigation/libtbus
./AAPP/src/navigation/libtle
./AAPP/src/preproc
./AAPP/src/preproc/bin
./AAPP/src/preproc/libatov
./AAPP/src/preproc/libatovin
./AAPP/src/preproc/libatovpp
./AAPP/src/preproc/libavh2hirs
./AAPP/src/preproc/libmappiasi
./AAPP/src/tools
./AAPP/src/tools/bin
./AAPP/src/tools/libaappbufr
./AAPP/src/tools/libaapphdf5
./AAPP/src/tools/libf7cp
./AAPP/src/tools/libf7gp
./AAPP/src/tools/libf7ml
./AAPP/src/tools/libf7nl1b
./AAPP/src/tools/libf7tp
./AAPP/src/tools/libsatid
./AAPP/src/tools_eccodes
./AAPP/src/tools_eccodes/bin
./AAPP/src/tools_eccodes/libecbufr
./config
./iasi-tools
./iasi-tools/bin
./iasi-tools/include
./iasi-tools/lib
./iasi-tools/man
./iasi-tools/src
./iasi-tools/src/bin
./iasi-tools/src/libcnes_iasi_brd_1.6
./iasi-tools/src/libcnes_iasi_brd_1.7
./iasi-tools/src/libcnes_iasi_ctx_1.2
./iasi-tools/src/libcnes_iasi_grd_1.6
./iasi-tools/src/libcnes_iasi_grd_1.7
./iasi-tools/src/libcnes_iasi_odb_1.4
./iasi-tools/src/libeps_iasil1b_6.6
./iasi-tools/src/libeps_iasil1b_9.0
./iasi-tools/src/libeps_iasil1c_6.6
./iasi-tools/src/libeps_iasil1c_9.0
./metop-tools
./metop-tools/bin
./metop-tools/include
./metop-tools/lib
./metop-tools/man
./metop-tools/src
./metop-tools/src/bin
./metop-tools/src/libaapp_avhrrl1b
./metop-tools/src/libaapp_avhrrl1c
./metop-tools/src/libccsds
./metop-tools/src/libeps_avhrrl1b_6.5
./metop-tools/src/libeps_common
./metop-tools/src/libeps_metop10
./metop-tools/src/libmetop_amsua
./metop-tools/src/libmetop_avhrr
./metop-tools/src/libmetop_common
```

./metop-tools/src/libmetop_hirs
./metop-tools/src/libmetop_intex
./metop-tools/src/libmetop_mhs
./metop-tools/src/libmetop_mmam
./metop-tools/src/libobtutc

In the run-time directory $AAPP_PREFIX (if applicable) you will find:
./AAPP/
./AAPP/bin/
./AAPP/lib/
./AAPP/include/
./AAPP/data/
./metop-tools/
./metop-tools/bin/
./metop-tools/lib/
./metop-tools/include/
./iasi-tools/
./iasi-tools/bin/
./iasi-tools/lib/
./iasi-tools/include/

TLE files for the test cases are provided as part of the test case. For running with your own HRPT data you should create the following directories: ${DIR_NAVIGATION}/tle_db and/or ${DIR_NAVIGATION}/tbus_db (where DIR_NAVIGATION is defined in ATOVS_ENV8) and place your TBUS or TLE files in monthly subdirectories. In the case of TLE, the "get_tle" script does all this automatically, creating directories as required.

After installation

After you have executed the installation procedure a few steps have to be performed before you can go to run AAPP:

- Check whether your installation was successful. Note that the error messages from the Fortran compilations might differ depending on the compiler you are using. Some compilation warnings are to be expected, but you should not see any actual errors. The "make" process will terminate early if there is an error.
- If you have customised the configuration file (e.g. are using a different compiler from those that are supported), it is advisable to run the tools det_ftnfort and det_reclen (both installed into AAPP/bin) to check that you have used the correct convention for Fortran file names and record length specifiers. These scripts will tell you what to do if the values need changing.
- You should be aware that l1b/l1c/l1d output files will always be in the native byte order of the machine you are running on (not applicable to BUFR output, which is character-oriented). If you're not sure whether your machine is big or little endian, type a command such as echo 01 | od -x . The result will be 3130 on a little-endian system and 3031 on a big-endian system.
- If your UNIX system does not have a Korn shell then you will need to adapt the various scripts (e.g. in directory AAPP/bin) to your shell type. Modify also accordingly the *.ksh scripts contained in subdirectories of AAPP/src in order not to lose the modifications when you rebuild AAPP in order to implement e.g. code modifications.
- Some of the AAPP commands have man pages. Add directory (destination directory)/AAPP/man/ to your MANPATH list to get access to the AAPP manual pages by e.g. adding to your .profile

  MANPATH=(destination directory)/AAPP/man:$MANPATH

  The man pages are contained in the subdirectories man1/, man3/ and man5/ of directory (dest.-dir.)/AAPP/man. You can display the man pages in the usual way by typing at your shell prompt

  man filename (without .suffix).

  However, the definitive descriptions of the AAPP commands are given in the software description document, rather than the man pages. And not all commands have a man page.

- To allow you to call AAPP scripts and binaries directly, it is recommended that you source the ATOVS_CONF file as part of your own setup, as described previously.
- Orbital prediction: Two orbital prediction methods are implemented in AAPP v8: TBUS and Two-Line-Element (TLE). A third method, referred to as the "SPOT model", is now obsolete. TLE is recommended as an alternative to TBUS since the accuracy is usually better. The data may be downloaded from the Space-Track web site using AAPP script *get_tle*, or alternatively from Celestrak. You will need to obtain a user name and password for the Space-Track site, which should be entered in ATOVS_ENV8 under parameters PAR_NAVIGATION_TLE_USER and PAR_NAVIGATION_TLE_PASSWD. This assumes that the computer on which you run AAPP has internet access; if it does not then you will need to download the files using a different computer and transfer them. To choose which method to use for a given satellite, just modify the ATOVS_ENV8 variable PAR_NAVIGATION_LISTEBUL, e.g.

  ```
  PAR_NAVIGATION_DEFAULT_LISTESAT='noaa19 noaa18 noaa15 M02 M01'
  PAR_NAVIGATION_DEFAULT_LISTEBUL='tle tle tle tle tle'
  ```

  AAPP will use TBUS bulletins for satellites for which "tbus" is specified and TLE for satellites with "tle". These parameters are used by commands *AAPP_RUN_NOAA, AAPP_RUN_METOP, alleph, amsuacl, amsubcl, hirscl, mhscl* and *avhrcl*. These programs are not applicable to the NPP satellite, and therefore NPP is not included in the `PAR_NAVIGATION_DEFAULT_LISTESAT` list. But you can generate satpos files for NPP in the usual way, by running tleing -s NPP and satpostle -s NPP.

  There are 4 other optional environment parameters related to TLE – *get_tle* will provide default values if they are not defined by the user or in ATOVS_ENV8. The defaults are given below:

  ```
  export DIR_DATA_TLE=${DIR_NAVIGATION}/tle_db  #directory to store data
  export PAR_NAVIGATION_TLE_TIMEOUT=60 #connection timeout in seconds
  export PAR_NAVIGATION_TLE_URL_DOWNLOAD='https://www.space-track.org'
  export PAR_NAVIGATION_TLE_CATALOGUE='25338,26536,27453,28654,33591,37849,29499,38771,27431,32958,37214,25994,27424' #Catalogue numbers
  ```

- Choice of HIRS calibration algorithm:
  The HIRS calibration algorithm is selected by means of environment variable HIRSCL_VERSION in ATOVS_ENV8. To select the old algorithm, as used in AAPP v4 and earlier, set the value to 0. To select the newer algorithm set it to 1. The new algorithm accumulates data on calibration runs from previous passes, and is designed to give better consistency with global data, especially for partial super-swaths. There are various other parameters available for the new algorithm – see comment lines in ATOVS_ENV8. If in doubt, use the default for these.
- Locale
  On some systems the unix "sort" command (which is used by several AAPP scripts) may give unexpected answers. You should check your locale settings (type locale): you should see LC_ALL=C or LANG=C (with LC_ALL undefined). If this is not the case, add LC_ALL=C to your shell startup script (or add it to ATOVS_ENV8).

A note on the Community Satellite Processing Package (CSPP) and HDF utilities

The CSPP is supplied by the University of Wisconsin, for level 1 and level 2 processing Suomi-NPP and JPSS direct readout data. For detailed information, see the CSPP web page. A detailed description of CSPP is beyond the scope of this document. However, it is worth pointing out some matters relevant to AAPP.

Firstly, CSPP includes pre-built copies of various libraries, including HDF5 library (C interface – not Fortran90), SZIP, Jasper, etc. You may wish to link to these when building AAPP.

Secondly, CSPP includes the HDF5 tools, which can be used to complement AAPP. These are in directory CSPP2.2/SDR_2_2/common/local/bin/ . You may wish to add this directory to your $PATH. Of particular relevance is the utility nagg, which can be used to concatenate NPP products and granules. For example, to concatenate granules of VIIRS channels M9, M10 and the terrain-corrected geolocation:

```
nagg -g 0 -n 100 -O cspp -D dev -t GMTCO,SVM09,SVM10 GMTCO*.h5 SVM*.h5
```

For more detail on this utility, see the corresponding HDF group web page.

Testing

Test cases

After having successfully installed AAPP on your workstation you may start testing AAPP. Several different test cases are provided. You should run the ones that are relevant to your application. The test cases supplied with AAPP v8.1 are listed in the following table. Other test cases may be available via the AAPP web pages. Each test case is provided with a 00README.txt file that gives detailed instructions.

Title	File	Purpose	Scripts	Notes
NOAA-19 HRPT	noaa19_test.tgz	Processes raw HRPT to level 1d for AMSU-A, MHS and HIRS, and AVHRR to level 1b. Optionally run MAIA to generate cloud masks on the HIRS grid and/or the AVHRR grid. Data from July 2017.	noaa19_run.sh noaa19_maia_avhrr.sh	No external libraries needed for the basic l1c/l1d generation. To run MAIA, set environment variable RUN_MAIA=”yes”. HDF5 library needed for MAIA. GFS forecast files are downloaded from the internet.
Metop-B AHRPT	metopb_test.tgz	Processes raw HRPT to level 1d for AMSU-A, MHS and HIRS, and AVHRR to level 1b. Optionally run OPS-LRS for IASI. Optionally run MAIA to generate cloud masks on the HIRS grid and/or the AVHRR grid. Data from July 2017.	metopb_run.sh metopb_maia_avhrr.sh	No external libraries needed for the basic l1c/l1d generation. To run OPS-LRS, set environment variable PATH_OPS. Auxiliary files are downloaded from NWPSAF web site if they are not already present. To run MAIA, set environment variable RUN_MAIA=”yes”. HDF5 library needed for MAIA. GFS forecast files are downloaded from the internet.
ATMS/CrIS preprocessing	atmscris_test.tgz	Ingest SDR data files generated by CSPP. Carry out spatial filtering for ATMS and spectral thinning for CrIS. Map ATMS to CrIS.	atmscris_run.sh	HDF5 library needed.
BUFR	bufr_test.tgz	Exercise the BUFR encode/decode routines for both BUFRDC and ecCodes, and compare the results. Uses the 1c/1d output files from the NOAA19, Metop-B and ATMS/CrIS test cases, and also a set of SDR files for the FY-3C instruments.	bufr_run_bufrdc.sh bufr_run_eccodes.sh compare_bufr_data.sh compare_decoded_data.sh	BUFRDC and/or ecCodes library needed. HDF5 library needed for FY-3C ingest.

There are also some legacy test cases on the AAPP web pages that are still relevant for AAPP v8:

Title	File	Purpose	Scripts	Notes
Metopizer	metopizer_and_AAPP.tgz	Processing of Metop CADU data to level 1c	run_metopizer_and_AAPP.sh	Metopizer is a software package supplied by EUMETSAT, for handling raw AHRPT. Available from the Support Software and Tools page.
VIIRS to CrIS	viirs_to_cris.tgz	Mapping of VIIRS imager data to CrIS footprint	viirs_to_cris_run.sh viirs_to_cris_withMAIA4.sh	2 granules of CrIS data and 1 granule of VIIRS data are supplied. Makes use of “nagg”. For MAIA4, forecast files are downloaded from the internet.
MMAM	test_MMAM.tgz	Extraction of Multi Mission Administrative Message from Metop HKTM file	test_mmam.sh	

Each test case is contained within a zipped tar file. To unpack a test case, copy the file to a suitable directory and type tar -xzf name.tgz

Before running any of the test cases, you should set the environment variable AAPP_PREFIX to the installation directory for AAPP v8.

Preparation for Processing Your Own HRPT Read-Out Data

Before trying to use AAPP for processing NOAA HRPT read-out data from your own station check first whether

- your station delivers the data in the format expected by AAPP, i.e. the 10-bit words of the down-linked HRPT data are stored right-justified in 16-bit words.
  If this is not the case you have to reformat the data. If your data are in packed 10-bit words then you can use the tool unpack_noaa_hrptto unpack the data, but you will probably need to change the values of parameters bytes_in and words_out in unpack_noaa_hrpt.F, since different stations use different conventions.
- your station position is contained in AAPP/src/navigation/libnavtool/stations.txt. If this is not the case insert it and copy the updated file additionally to AAPP/data/navigation/, and modify the variable “STATION” in the environment variable file ATOVS_ENV8 with the name of your station. Note: if “STATION” is empty then AAPP_RUN_NOAA will fail!

To process METOP AHRPT data you must make sure that your station produces PFS Level 0 data files, as defined by EUMETSAT. If the reception station does not deliver files in L0 format then the user should consider using METOPizer software to pre-process the raw AHRPT (go to www.eumetsat.int and navigate to Data -> Data Delivery -> Support Software & Tools). The METOPizer programs tvcdu_to_ccsds and ccsds_to_l0 are likely to be needed. Please note the following advice from EUMETSAT:

If you are reading CADU packets then the installation procedure has changed. The Metopizer used to include a Reed-Solomon library licensed

under the GPL which has to be installed separately now. There is an INSTALL file inside the package with instructions.

To process NPP and JPSS data you will need an external processing package to generate Sensor Data Record (SDR) files suitable for input to AAPP. Suitable packages are supplied by NASA ("IPOPP") and by the University of Wisconsin ("CSPP"). Details are available elsewhere.

Processing scripts

Direct readout processing
The AAPP_RUN_NOAA chain script processes NOAA HRPT data through to level 1d. The script offers the following options:

- Specify the instruments to be processed (from the list "AMSU-A AMSU-B HIRS MSU AVHRR DCS ATMS CRIS")
- Specify the output grids required at level 1d (from the list "AMSU-A AMSU-B HIRS ATMS CRIS")
- Specify an output directory to receive the product files
- Omit avh2hirs if not required

In early versions of AAPP, it was suggested that the user should customise the script to suit his own requirements, but now that the functionality is provided by command-line options, this should be unnecessary. If you do wish to customise the script, please give it a different name, to avoid confusion.

If your HRPT reception system automatically delivers the orbit number, time, date, etc. (e.g. in the file name) then you may pefer to use a simplified script, as in AAPP_RUN_NOAA_simplified.

In the example below we use environment variables "ATOVS" and "AVHRR" to control which parts of AAPP are run.

To process METOP AHRPT data you may use the AAPP_RUN_METOP script which works as follows:

- Looks for PFS level 0 files in an input directory (dump mode assumed, i.e. one file per instrument)
- Processes data for the specified instruments (from the list "AMSU-A MHS HIRS AVHRR IASI")
- Generates 1d output on the requested grids ("AMSU-A MHS HIRS IASI")
- Sends output files to the specified output directory

The script allows the user to take two passes through the data – the first to generate traditional HIRS, AMSU-A and MHS 1d files, and the second optional pass to run OPS-LRS and generate products on the IASI grid. For full details, please see the description in the source.

If your station does not generate one file per instrument per pass (e.g. it uses 3 minute granules) then you will need to concatenate the granules, otherwise the HIRS calibration will fail. Level 0 granules can beconcatenated using the Unix "cat" command.

Processing externally supplied level 1b files
Many users also use AAPP to process global or regional ATOVS data, which are typically supplied at level 1b or 1c. To do this you need to call atovin and/or atovpp directly from your script. For example, to convert ATOVS data from level 1b to level 1d on the HIRS grid you would typically do the following:

```
ln -sf amsua_1bfile aman.l1b
ln -sf amsub_1bfile ambn.l1b
ln -sf hirs_1bfilehrsn.l1b
atovin AMSU-A AMSU-B HIRS
atovpp -i "AMSU-A AMSU-B HIRS" -g "HIRS"
```

Note that if you are ordering data from NOAA CLASS, you should untick the box that says "include archive header" (or do this in your user preferences). Otherwise you will have to strip off the first 512 bytes from each 1b file before running atovin (e.g. dd bs=512 skip=1).

To process historic TOVS level 1b data from CLASS (i.e. NOAA-14 and earlier), you have to first run the script noaa_class_to_aapp.

In the case of METOP data distributed in BUFR, you would need to run aapp_decodebufr_1c and then atovpp.

But note that global METOP data are distributed in short (typically 3 minute) granules, and there is no guarantee that the granules for the different instruments will be aligned in time. Therefore a utility has been provided to concatenate level 1c files, called combine_1c, in order to ensure that the instrument to be mapped covers a wider time interval than the instrument to be used as a grid. It is up to the user to write his script such that the appropriate input files are concatenated.

If difficulties are encountered, please seek advice using the Feedback Form available from the NWP SAF.

Processing ATMS and CrIS data
ATMS and CrIS data may be received as near-real-time global BUFR files or as SDR files. You will need to run some or all of the following AAPP tools:

- aapp_decodebufr_1c, to convert from BUFR to AAPP-style level 1c
- atms_sdr or cris_sdr, to convert from SDR in HDF5 format to AAPP-style level 1c
- atms_beamwidth, to modify the effective beamwidth and noise characteristics of ATMS (at level 1c)
- atovpp, to create a CrIS level 1d file (ATMS + CrIS, various options) or to create an ATMS level 1d file on the original ATMS grid
- cris_channels is for users who need to generate a channel-selected 1c dataset, independent of atovpp. For example, DBNet stations can use it during the conversion from SDR to channel-selected BUFR product. Uses the channel selection in CRIS.fdf. Note that AAPP convention is to use the standard channel numbering system, where channel 1 is 650 cm^{-1} and there are 1305 channels. (There are additional channels for "Full Spectral Resolution" mode)
- atms_thin1d, to create a spatially thinned ATMS product (AMSU-like sampling)
- aapp_encodebufr_1c
  - BUFR encode the 1c ATMS or CrIS files, or
  - Create a BUFR-encoded level 1d file for ATMS or CrIS. This option was developed for use at the Met Office (and uses some local BUFR descriptors), but may be useful to others.

For more information on the processing, see AAPP document NWPSAF-MO-UD-027 "Pre-processing of ATMS and CrIS".

In some cases you will need to do some preparation of the data before running the AAPP utilities. Data often arrive with one file per 32-second granule – which is too short for effective use of atms_beamwidth and atovpp. Also, in some cases the granules may not arrive in time order. One solution to this problem that has been employed at the Met Office for ATMS/CrIS BUFR data is to aggregate files as follows:

1. Use the time stamp in the file name to compute a granule number, defined as the second of day divided by 32. Append this to the file name.
2. Also compute an aggregation number, equal to the granule number divided by the number of granules you wish to aggregate (e.g. 10). Append this to the file name also.
3. When all 10 granules have arrived for a given aggregation number, concatenate the individual granules (unix cat will work for BUFR data) and proceed to the AAPP processing.

A similar approach could be used for processing HDF5 granules, except that atms_sdr or cris_sdr would be run first, and the *.l1c granules aggregated using combine_1c.
This problem does not arise with HDF5 data from NOAA/CLASS because files can be delivered already aggregated into chunks of 8 minutes (15 granules).

Processing archived data from EUMETSAT Data Centre
AAPP can ingest the following types of data from the EUMETSAT Data Centre:

- EPS level 0
- EPS level 1 (native format) for AVHRR or IASI (but not for AMSU, MHS or HIRS)
- BUFR level 1 for AMSU, MHS, HIRS, IASI

Note that unless you specify a regional subset, then the input files will be full-orbit. For AVHRR and IASI this results in quite large input files. If you wish to process AVHRR files that contain more than 33 minutes of data, the following changes are needed in AAPP:

1. For level 0 processing, increase the value of mx_hrpscn in hrptdc.h. Normally set to 12000; increase to 32767 which is the maximum possible, since the AVHRR 1b format holds avh_h_scnlin as a 2-byte integer. Note that a full-orbit would be 36400 scans.
2. Increase the value of avh_mxscn in avhrcl.h. Also normally set to 12000.

It is preferable to specify a geographical subset (defined by lat/lon limits) when you order the data.

The applicable AAPP scripts are:

- AAPP_RUN_METOP, for processing level 0. You will need to obtain appropriate orbital elements files.
- convert_avh1b, to convert AVHRR EPS level 1b to AAPP 1b format – but with scaled radiances instead of the original instrument counts
- convert_iasi1c, to convert IASI EPS level 1b to AAPP 1c format
- aapp_decodebufr_1c or eccodes_decodebufr_1c, to decode BUFR files

Satellite attitude

Some users may wish to manually set the values of roll, pitch and yaw ("attitude") for a particular satellite. The default attitude for each satellite is set in the file ${DIR_NAVIGATION}satid.txt. Look for the lines that say "number of orbital bulletin dependant attitude biases". Underneath are listed the default yaw, roll and pitch (in milliradians) for up to four types of bulletin (in the order TBUS, Argos, 2-line, Spot). These values may be changed by the user if required, or attitude biases inserted if there are no values already present (i.e. older satellites). This file also specifies misalignment parameters for individual instruments.

If you wish to change the attitude for specific orbits then you need to create a file ${DIR_NAVIGATION}/ana/estatt_${BUL}_${SATIMG}.txt, where ${BUL} is "tbus" or "tle" and ${SATIMG} is the name of the satellite, e.g. "noaa19". This is a text file with the following values on each line

yaw roll pitch orbit 0

where yaw, roll and pitch are in milliradians and "0" means the data are good. If this file exists, it is read by the scripts avhrcl, amsuacl, amsubcl, etc. If the orbit being processed (or the previous orbit) matches a line in the file then the corresponding attitude values are used; if not then the default (from satid.txt) is used.

Satellite manoeuvres

Some satellites (notably MetOp) undergo pre-planned manoeuvres in order to maintain the correct orbit. For MetOp these may be either "in-plane" (no change to attitude) or out-of-plane (in which the satellite is rotated so that the station-keeping rockets are pointing at 90 degrees to the direction of motion). For AAPP users the in-plane manoeuvres are usually of little concern, however the out-of-plane manoeuvres (conducted typically every 1-2 years) can cause problems with the quality control of the orbital elements files (TBUS or 2-line).

When an out-of-plane manoeuvre is announced it is recommended for users to temporarily disable the quality control mechanisms by setting an environment variable as follows:

PAR_NAVIGATION_EXTRAP_LIMIT=100
export PAR_NAVIGATION_EXTRAP_LIMIT

This may be either inserted into your ATOVS_ENV7 file or into your calling script. When the manoeuvre is finished, and new orbital elements have been ingested, then you can set PAR_NAVIGATION_EXTRAP_LIMIT="".

To check whether a bulletin has been successfully ingested, look at your index file, e.g. $DIR_NAVIGATION/tle_db/tle_M02.index. The second number in each line should be "0" if the corresponding bulletin was OK. To ingest a specific TLE file by hand (e.g. published in the MetOp Admin message) then you can run tleing by hand with the "-f" option (type tleing -h for usage instructions), e.g.

tleing -s M02 -f ./2011-10/tle_20111005_1130.txt

If you manually ingest a bulletin in order to correct a geolocation error, remember to delete the old satpos file, otherwise AAPP will continue to use that satpos file for the remainder of the day and you will still get the error.

On the use of working directories

The ATOVS_ENV8 setup defines an environment variable $WRK (default $HOME/tmp), which is used by several of the AAPP scripts. For example, AAPP_RUN_NOAA and AAPP_RUN_METOP do a "cd" to this directory during the script execution. Other scripts assume you are already in a suitable working directory (e.g. atovpp). You can set up your working directory before running the script, e.g. to use the current directory do

```
WRK=$PWD && export WRK
```

If you are running multiple instances of AAPP on the same machine (e.g. processing global and local data at the same time), it is very important that they do not attempt to use the same working directory, otherwise you will get unpredictable results.

One way to do this would be to define temporary working directories before running a script, e.g.

```
WRK=$PWD/tmp_$$_$(date +%H%M%S_%N) && export WRK
mkdir -p $WRK
cd $WRK
```

where the temporary directory has been created accoring to the shell process number ($$) and the system time. Remember to delete the temporary directory when you have finished with it.

Most users are unlikely to need this level of complexity, but the issue has been raised in the past via the NWP SAF Helpdesk, therefore we mention it as something to be aware of when you are designing your processing systems.