NWP SAF

AAPP-SG User Manual

Version: 1.0

26 May 2026

AAPP-SG User Manual

This documentation was developed within the context of the EUMETSAT Satellite Application Facility on Numerical Weather Prediction (NWP SAF), under the Cooperation Agreement dated 7 September 2021, 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
0.1	27/11/2025	Nigel Atkinson	Initial draft
0.2	16/02/2026	Nigel Atkinson	
0.3	06/03/2026	Elisabeth Nolland	
0.4	13/04/2026	Nigel Atkinson	For external review
1.0	26/05/2026	Elisabeth Nolland	Following external review

Table of Contents

1. INTRODUCTION ... 4

1.1 Reference documents ... 4
1.2 Software scope ... 4
1.3 Functionality ... 5
1.4 Data availability ... 6
1.5 System requirements ... 6

2. INSTALLATION ... 7

2.1 Download and unpack AAPP-SG ... 7
2.2 Quick installation recipe: build from source ... 8
2.3 Download and install ecCodes, hdf5 and netCDF ... 9
2.3.1 Required library versions ... 9
2.3.2 Installing libraries via script ... 9
2.3.3 Installing libraries via conda ... 10
2.4 Install AAPP-SG from source ... 10
2.4.1 Configure AAPP-SG ... 11
2.4.2 Build and install AAPP-SG ... 11
2.5 Install AAPP-SG from binaries ... 12
2.5.1 OS compatible with RHEL9 ... 12
2.5.2 OS incompatible with RHEL9 ... 13

3. RUNNING THE SOFTWARE ... 13

3.1 Environment ... 13
3.2 Namelists ... 14
3.3 IASI-NG processing ... 14
3.3.1 NG_main.exe ... 15
3.3.2 NG_1c_to_1d.sh ... 18
3.3.3 NG_netcdf_to_bufr_1c.exe ... 18
3.3.4 NG_netcdf_to_bufr_1d.exe ... 19
3.4 MWS processing ... 19
3.4.1 mws_main.exe ... 19
3.4.2 mws_netcdf_to_bufr_1b.exe ... 21
3.5 Display utilities ... 21

4. TEST CASES ... 22

5. SCIENTIFIC DESCRIPTION ... 23

5.1 IASI-NG Principal Components manipulations ... 23
5.2 Conversion to IASI first generation spectral response ... 24
5.3 IASI-NG spatial thinning ... 25
5.4 MWS spatial filtering ... 26
5.5 MWS scattering and surface tests ... 27
5.5.1 Scattering tests ... 28
5.5.2 Surface test ... 28

6. BUFR SEQUENCES ... 29

6.1 IASI-NG level 1C ... 29
6.2 IASI-NG level 1D ... 32
6.3 MWS level 1B ... 33

1. INTRODUCTION

This document is the user manual for the AAPP pre-processing module for new generation of polar satellites (starting with EPS-SG). The software is referred to as AAPP-SG.

Currently, AAPP-SG is dedicated to the Metop-SG-A series of satellites. Other satellites may be added at a later date.

The document incorporates the following elements:

- Installation Guide (Section 2)
- Run-time instructions (Section 3)
- Test cases (Section 4)
- Scientific description (Section 5)
- List of BUFR sequences (Section 6)

The Digital Object Identifier (DOI) of the package is: **10.15770/EUM_SEC_CLM_1012**

1.1 Reference documents

RD-1: NWPSAF-MO-DS-044: Design proposals for EPS-SG pre-processing in AAPP and MWIPP

RD-2: NWPSAF-MO-DS-045: AAPP v9 Product Specification

RD-3: NWPSAF-MO-DS-046: AAPP v9 Top Level Design

RD-4: NWPSAF-MO-TR-040: Technical report: Adapting the AAPP microwave tests for EPS-SG

RD-5: NWPSAF-MO-TR-041: Technical report: Adapting the AAPP microwave tests for EPS-SG: part 2 – surface tests

Documents RD-1 to RD-5 are available on-line at
https://nwp-saf.eumetsat.int/site/software/aapp/future-plans/

RD-6: NWPSAF-MO-UD-027: Pre-processing of ATMS and CrIS, available at https://nwp-saf.eumetsat.int/site/software/aapp/documentation/

RD-7: NWPSAF-MO-UD-053: IRSPP User manual, available at https://nwp-saf.eumetsat.int/site/software/irspp/

1.2 Software scope

As mentioned previously, this document describes the pre-processing module for Metop-SG. At the time that RD-1, RD-2 and RD-3 were first formulated, it was planned that AAPP as a whole would be upgraded to version 9, with the Metop-SG pre-processor implemented as a standalone module within AAPP – formally part of AAPP, but not dependent on the legacy components.

However, for operational reasons it has been decided to release the software as a standalone package while the legacy parts of AAPP – now to be referred to as AAPP-core - are still at v8. As AAPP-SG has no dependency on AAPP-core, this has no impact on the functionality of the software.

In this document the Metop-SG pre-processor is now given the name ***AAPP-SG*** – to reflect the fact that it is a pre-processor for EPS-SG and other new-generation polar satellites, while the legacy parts of AAPP are referred to as AAPP-core.

The main purpose of AAPP-SG is to prepare the level 1 data (whether global or direct broadcast) for use in NWP, through various data transformations and format conversions.

It is also important for users to realise what is outside the scope of AAPP-SG:

- AAPP-SG does not process raw or level 0 direct broadcast data. That is the function of the EPS-SG level 1 direct broadcast packages that have been procured by EUMETSAT: they are tested and distributed by the NWP SAF but were not developed by the NWP SAF, and are delivered outside the AAPP framework[1].
- AAPP-SG does not do any dedicated processing for METimage; users requiring a METimage cloud mask should use the PPS software developed by the NWC SAF[2].
- AAPP-SG does not process level 2 data.

1.3 Functionality

In AAPP-SG, the following definitions are used for data levels:

- Level 1b (applicable to MWS): calibrated brightness temperatures as measured by the instrument
- Level 1c (applicable to IASI-NG): apodised radiances on a standard spectral grid
- Level 1d: radiances or brightness temperatures have been subjected to additional processing steps, e.g. spatial or spectral thinning, re-mapping from one instrument grid to another, PC scores instead of raw radiances, computation of reconstructed radiances

AAPP-SG has the following functionality:

- Ingest level 1b data from MWS, in netCDF or BUFR format.
- Ingest level 1c data from IASI-NG, in netCDF or BUFR format
- Ingest level 1d data (with PC scores) from IASI-NG, in netCDF or BUFR format .
- Carry out spatial filtering and thinning for MWS.
- Convert from level 1c to level 1d for IASI-NG, i.e. compute the PC scores, select specified channels and/or carry out spatial thinning
- Re-map MWS onto the IASI-NG spatial grid
- Create BUFR outputs at level 1b (MWS), level 1c (IASI-NG) or level 1d (MWS and IASI-NG). The BUFR sequences are detailed in Section 6. In the case of IASI-NG level 1d BUFR, two variants are provided: (i) a standard level 1d BUFR sequence (WMO approved) and (ii) appended with additional information on IASI-NG radiances and mapped MWS brightness temperatures (referred to as “enhanced level 1d” in this document).

[1] https://nwp-saf.eumetsat.int/site/metop-sg-direct-broadcast-software/
[2] https://www.nwcsaf.org/

1.4 Data availability

It is helpful to put AAPP-SG into the context of near-real-time data availability. See the summary in Table 1.

Table 1: Summary of the operational level 1 products from MWS and IASI-NG

Product	NRT availability	Main relevance to AAPP-SG
MWS level 1b netCDF	Output from DB package	DBNet stations and other DB users will use the software to convert to BUFR, e.g. for international dissemination
MWS level 1b BUFR	EUMETCast Satellite and Terrestrial; WIS2.0	Users perform spatial filtering and thinning for NWP, converting to L1d BUFR
IASI-NG level 1c netCDF	Output from DB package	DBNet stations and other DB users will use the software to convert to BUFR
IASI-NG level 1c BUFR (full spectra)	EUMETCast Terrestrial; WIS2.0	Some NWP users will prefer this product to the PC product.
IASI-NG level 1d BUFR (PC scores)	EUMETCast Satellite and Terrestrial; WIS2.0	Users will generate reconstructed radiances for their required channels.

1.5 System requirements

AAPP-SG is available in the form of source code or binaries built on a Red Hat Enterprise Linux (RHEL) 9 operating system.

To build, install and run AAPP-SG from source code, as documented in Sections 2.1 to 2.4, you need the following:

- A Linux system, compatible with RHEL9
- A Fortran-90 compiler that supports Fortran-2003, e.g. gfortran or ifort
- ecCodes library (see Section 2.3)
- If you are building ecCodes, cmake is required
- hdf5 library
- netCDF C and Fortran libraries
- gmake or equivalent

To run AAPP-SG from the binaries, as per Section 2.5, you need the following:

- A Linux system, compatible with RHEL9
  OR
- Access to Apptainer

These are essentially the same requirements as other NWP SAF deliverables such as AAPP-core, IRSPP and MWIPP.

If you are not sure what versions of the external libraries are installed on your system, try running commands such as:

```
bufr_dump -V
h5dump -V
```

```
cmake –version
```

Note that recent ecCodes releases (v2.31.0 and higher) require a C++ compiler supporting the C++17 standard. If you use GNU compilers this means that you need at least gcc8.

AAPP-SG has been tested on systems compatible with 64-bit CentOS7 and CentOS9. Testing has been carried out against the following Fortran compilers:

- gfortran v11.5.0
- ifort v19.0.0

2. INSTALLATION

This section explains how to install AAPP-SG and its dependencies (ecCodes, hdf5, netCDF) assuming you do *not* have administrator privilege – i.e. the software packages are installed in local directories. If you are running as administrator, you should follow your local conventions.

There are three different approaches to installing AAPP-SG:

- Build AAPP-SG from source code and install the required dependencies using the supplied `install_dependencies.sh` script.
- Build AAPP-SG from source code and install the required dependencies using conda.
- Install AAPP-SG binaries.

Section 2.1 discusses where to find and download the AAPP-SG source code or binary files. Sections 2.2–2.4 describe source-code installation: Section 2.2 provides a quick copy-pasteable recipe for a typical installation using `install_dependencies.sh`, while Sections 2.3–2.4 provide more detailed instructions. These instructions include how to install only a subset of the dependencies, use pre-existing local or centrally installed libraries, or install the dependencies using conda. Finally, Section 2.5 explains how to install and run AAPP-SG from binaries.

2.1 Download and unpack AAPP-SG

The AAPP-SG package is available to download from the “Software Downloads” page on the NWP SAF website. To access this page, click on the “Software Downloads” link at the very top of the homepage. You will then be prompted to log in, if not already done.

Follow the instructions provided to change your Software Preferences to include the “AAPP-SG” package. The link to download AAPP-SG should then be included in the list of latest software package download links. You can choose whether to download the complete package, including source code, or whether to use pre-built binaries (built on RedHat 9).

The package is distributed as a compressed tar archive. To unpack the archive:

```
cd <directory in which to unpack and build AAPP-SG>
tar –xvzf <path to AAPP-SG tarfile>
```

Note that the source code is built in the same directory where it is unpacked.

If you are using binaries, please jump straight to Section 2.5.

2.2 Quick installation recipe: build from source

This section provides a simplified, copy-pasteable recipe to build AAPP-SG and all dependencies from source code. For a more customised approach, refer to the detailed instructions in Sections 2.2–2.4.

```
# Unpack the AAPP-SG source package
cd <directory in which to unpack and build AAPP-SG>
tar -xvzf <path to AAPP-SG source tarfile>

# Enter the unpacked source directory.
# AAPP-SG will be built in this directory.
cd AAPP-SG-X.X.X-build
export AAPP_SG_HOME=$PWD

# Optional: choose where dependencies will be installed.
# By default, dependencies are installed under:
#   ${AAPP_SG_HOME}/../NWPSAF_dependencies/install
# Uncomment and edit the following line to install elsewhere
# export AAPP_SG_DEPENDENCIES=/path/to/NWPSAF/dependencies

# Install all required dependencies
${AAPP_SG_HOME}/setup/install_dependencies.sh --all

# Optional: choose where AAPP-SG will be installed.
# By default, AAPP-SG is installed under:
#   ${AAPP_SG_HOME}/../AAPP-SG-X.X.X-install
# Uncomment and edit the following line to install elsewhere
# AAPP_SG_INSTALL=/path/to/AAPP-SG-install

# Configure AAPP-SG
if [[ -n ${AAPP_SG_INSTALL:-} ]]; then
  ${AAPP_SG_HOME}/setup/configure_aapp-sg.sh --prefix=${AAPP_SG_INSTALL}
else
  ${AAPP_SG_HOME}/setup/configure_aapp-sg.sh
fi

# Build and install AAPP-SG
cd ${AAPP_SG_HOME}/src
make
make install

# Set up the run-time environment
if [[ -z ${AAPP_SG_INSTALL:-} ]]; then
  AAPP_SG_INSTALL=${AAPP_SG_HOME}/../AAPP-SG-X.X.X-install
fi
```

```
source ${AAPP_SG_INSTALL}/bin/aapp-sg_env.sh
```

2.3 Download and install ecCodes, hdf5 and netCDF

2.3.1 Required library versions

AAPP-SG has the following build- and run-time library dependencies: hdf5, netCDF, and ecCodes.

Dependency	Recommended version/s	Minimum required version
hdf5	1.14.6	1.10.6
netCDF-C	4.9.3	4.8.1
netCDF-Fortran	4.6.2	4.5.3
ecCodes	2.44.0	2.44.0[3]

If all required dependencies are already installed (either centrally or in your local environment), you may skip ahead to Section 2.4. Otherwise, two installation methods are available: either install the required libraries using the provided installation scripts (Section 2.3.2), or install them using a package manager, such as conda, using the supplied requirements file (Section 2.3.3).

2.3.2 Installing libraries via script

To install missing dependencies, you can run the script `install_dependencies.sh`, which is in the `setup` directory of the untarred AAPP-SG package.

By default, this script creates a `NWPSAF_dependencies` subdirectory in a directory *above* that of the untarred AAPP-SG package, and installs the dependencies into `NWPSAF_dependencies/install`. The reason for putting it in a higher directory is to facilitate building using common dependencies across different versions of AAPP-SG or across multiple NWP SAF packages (e.g. IRSPP or MWIPP).

Alternatively, you can specify a different install location by exporting the environment variable `AAPP_SG_DEPENDENCIES` to point to your chosen directory, before running the script. In this case, the libraries will be installed to `${AAPP_SG_DEPENDENCIES}/install`.

This script can be used to install **any** or **all** of the following dependencies:

- CMake - Required to build ecCodes. The script checks whether a recent version of `cmake` is available in the user's PATH and only installs this package if not.
- HDF5 (with Fortran interfaces) - Required to build netCDF dependencies.
- netCDF-C - Required to build netCDF-Fortran.
- netCDF-Fortran
- ecCodes

[3] This version of ecCodes is required for supporting IASI-NG L1D BUFR encoding, see section 6.2. If you do not require this functionality, e.g. if only processing MWS, you may use an earlier version of ecCodes. The BUFR tables for MWS were introduced in v2.33.0.

The dependencies will be built using the GNU Fortran compiler (gfortran), by default. To use the Intel Fortran compiler (ifort) instead, set the environment variable:
`export FC=ifort`

To use a pre-existing HDF5 installation to build the netCDF packages, please read the instructions in the script header comments. Similarly, please read these instructions if you wish to use the `cmake` executable from a pre-existing CMake installation to build ecCodes.

For usage instructions, run: `./install_dependencies.sh --help`

<u>Typical usage</u>

- For HDF5 and netCDF libraries:
  `./install_dependencies.sh --hdf5 --netcdf-c --netcdf-f`
- For ecCodes library:
  `./install_dependencies.sh --cmake --eccodes`
- For HDF5, netCDF, and ecCodes libraries:
  `./install_dependencies.sh --all`

2.3.3 Installing libraries via conda

If you wish to use the conda package manager[4] to install the required dependencies, follow the steps below to set up the conda environment:

```
env_name=NWPSAF_dependencies # or your chosen name
env_file=/path/to/AAPP-SG source directory/setup/requirements.txt
conda create --override-channels --channel=conda-forge --name ${env_name}
--file ${env_file}
```

Then, to use this environment, run:

```
conda activate ${env_name}
```

before installing AAPP-SG from the source code (see Section 2.4).

Note that it is recommended that this environment is activated at run time. If not, it is essential that the environment still exists, otherwise AAPP-SG will not have access to required run time libraries.

2.4 Install AAPP-SG from source

We will refer to the untarred AAPP-SG directory as **AAPP_SG_HOME** in the following instructions.

The AAPP-SG installation process involves the following steps

1) `${AAPP_SG_HOME}/setup/configure_aapp-sg.sh <arguments, see Section 2.4.1>`
2) `cd ${AAPP_SG_HOME}/src && make`

[4] See documentation at https://docs.conda.io/projects/conda.

3) `make install`

Note that `make install` is optional but recommended. It also needs to be run from within the `src` directory.

It is strongly recommended that you use a "clean" environment when building AAPP-SG – i.e. representative of the environment that you will use when you run the software. Otherwise, the `aapp-sg_env.sh` (created by `configure_aapp-sg.sh`) may not contain all the necessary additions to PATH and LD_LIBRARY_PATH.

2.4.1 Configure AAPP-SG

The `configure_aapp-sg.sh` script populates `Makefile.ARCH` and `aapp-sg_env.sh` (and runs checks of your dependencies), in preparation for building AAPP-SG.

By default, this configure script ensures that AAPP-SG will be installed into:
`${AAPP_SG_HOME}/../AAPP-SG-<version>-install`
To specify an alternative installation directory, use the `--prefix` option.

For usage instructions, run: `configure_aapp-sg.sh --help`

Dependencies

- If the hdf5/ netcdf/ eccodes dependencies are available from the default location `${AAPP_SG_HOME}/../NWPSAF_dependencies/install`, this script will automatically configure AAPP-SG to use these dependencies.
- Alternatively, to use dependencies installed into a different local directory:
  `export AAPP_SG_DEPENDENCIES=/path/to/local/dependencies`
  Note that this directory **must** contain an `install` subdirectory where the packages have been installed.
- Individual dependencies can also be specified by the user by means of script arguments or environment variables, where script arguments take precedence.
- Otherwise, if you have built and activated a conda environment with the required dependencies (see Section 2.3.3), the script will automatically configure AAPP-SG to use the dependencies from this environment.
- If all else fails, the script will attempt to find and use centrally-installed dependencies.

Warning
Some systems may already have pre-defined environment variables such as $HDF5_LIB and $ECCODES_LIB that are unrelated to AAPP-SG. If this is the case, please unset them before running `configure_aapp-sg.sh`.

2.4.2 Build and install AAPP-SG

`make` creates and populates two new directories: `AAPP_SG_HOME/bin` and `AAPP_SG_HOME/build`.
`make install` copies `bin` and `definitions` from AAPP_SG_HOME to AAPP_SG_INSTALL .
Both steps use the settings in `Makefile.ARCH`, generated by the configure step.

The build and install directories generated by the installation should have the contents shown below.

- The AAPP-SG `build` directory structure should contain:

  `/setup` (contains `install_dependencies.sh` and `configure_aapp-sg.sh`)
  `/bin` (contains executables, and `aapp-sg_env.sh`)
  `/build` (contains generated modules and libraries)
  `/definitions` (BUFR tables)
  `/src`
     `/bin` (main programs)
     `/libmws`
     `/libiasi-ng`
     `/libmapping`
     `/libcommon`

- The AAPP-SG install directory (if used) should contain:
  `/bin` (contains executables, and `aapp-sg_env.sh`)
  `/definitions` (BUFR tables)

2.5 Install AAPP-SG from binaries

The binaries allow execution on an operating system compatible with Red Hat Enterprise Linux 9 (RHEL9). If your operating system is not compatible with RHEL9, use Apptainer to create an environment in which the binaries can be run, see Section 2.5.2.

2.5.1 OS compatible with RHEL9

Save the AAPP-SG_vX.X.X-exec.tgz file to a suitable directory, unpack it and set up your environment as follows:

```
tar -xzf AAPP-SG_vX.X.X-exec.tgz
cd AAPP-SG-X.X.X-exec
export AAPP_SG_INSTALL=$PWD
source ${AAPP_SG_INSTALL}/bin/aapp-sg_env.sh
```

Setting AAPP_SG_INSTALL is mandatory when using the binaries. The aapp-sg_env.sh performs three functions:

1. Adds `$AAPP_SG_INSTALL/bin` to the front of your PATH (if it is not already there)
2. Adds `$AAPP_SG_INSTALL/lib` to the front of your LD_LIBRARY_PATH
3. Defines ECCODES_DEFINITION_PATH and ECCODES_SAMPLES_PATH to access the local BUFR tables and the default tables and samples.

This should allow the executables to be run. You will be able to control the functionality of AAPP-SG using namelists (see following sections) but of course you will not be able to make any changes to the code.

In order to ensure that the executables can be run with no dependencies, the ecCodes `share` directory has been included in the tar file. If you have your own installation of ecCodes and prefer to use those tables, then simply adjust the environment variables in `aapp-sg_env.sh` so that they point to your own ecCodes shared files.

2.5.2 OS incompatible with RHEL9

If you are not running on an OS compatible with RHEL9, you may be able to use ***Apptainer*** to create an environment in which the binaries can be run. For example, the following commands will create a Rocky Linux 9 sandbox in a directory ${DIR} and enter it interactively:

```
apptainer build --sandbox ${DIR}/rocky9/ docker://rockylinux:9
apptainer shell ${DIR}/rocky9
```

Then define AAPP_SG_INSTALL and source the `$AAPP_SG_INSTALL/bin/aapp-sg_env.sh` as described previously. On most systems this can all be done without needing root privilege.

If you prefer to run AAPP-SG from a top-level wrapper script (i.e. instead of entering the sandbox interactively), you can invoke it using, e.g.

```
apptainer run ${DIR}/rocky9 ./myscript.sh
```

Note that the `$AAPP_SG_INSTALL/lib` directory contains all the dynamic libraries that are required to supplement those provided by default by Apptainer. So there should be no need to install any dependency libraries.

3. RUNNING THE SOFTWARE

3.1 Environment

In this section, ${AAPP_SG_INSTALL} is the full path of the installation directory, i.e. the directory above the `bin` directory where the executables reside.

It is recommended to source the environment file `aapp-sg_env.sh` before running any main program, in order to set up your PATH, LD_LIBRARY_PATH and ECCODES_DEFINITION_PATH

```
source ${AAPP_SG_INSTALL}/bin/aapp-sg_env.sh
```

As mentioned in Section 2.4.1, having set up the environment, you should *not* re-run `configure_aapp-sg.sh`, otherwise your `aapp-sg_env.sh` file will be incorrect.

3.2 Namelists

The Fortran executables rely on namelists for the user to control the various options. The general structure of a namelist is as follows (where "title" has to match that assigned in the equivalent Fortran namelist definition). In the tables below, the title is in each case shown in red.

```
&title

variable1 = value1
variable2 = value2
[variableN = valueN]

/
```

If any variable is omitted, the default (where provided) is used.

The software design is such that any quantities that the user may want to change frequently (e.g. input and output file names) are passed as arguments, while the quantities that are more likely to be fixed (such as the names of ancillary files) are held in namelists. In some cases the name of the top-level namelist is passed as an argument – since the users may want to use different top-level namelists for different purposes (e.g. one for processing IASI-NG level 1C, another for processing level 1D).

For examples of working namelists, please refer to the test cases, see Section 4.

3.3 IASI-NG processing

The top-level routines for IASI-NG processing are listed in Table 2.

Table 2: Top-level routines for IASI-NG

Executable	Functionality
NG_main.exe	• Read input file in netCDF or BUFR (1c or 1d) • Perform channel selection (1c or 1d) • Compute reconstructed radiances (1d) • Convert to IASI first generation spectral response (1c or 1d) • Map MWS to IASI-NG grid (if a co-located MWS file is supplied) • Write output as BUFR: 1c, 1d or enhanced 1d. The enhanced 1d includes radiances (raw or reconstructed) for the channel selection and also the mapped MWS brightness temperatures. • If the input file is netCDF, there is the option to append information in a "derived" group
NG_1c_to_1d.sh	Shell script to convert level 1c to level 1d netCDF. It first invokes netCDF utility ***nccopy***, then calls executable *NG_1c_to_1d.exe*
NG_netcdf_to_bufr_1c.exe	netCDF to BUFR converter for level 1c
NG_netcdf_to_bufr_1d.exe	netCDF to BUFR converter for level 1d

3.3.1 NG_main.exe

Usage:

```
NG_main.exe [-n namelistfile] [-i infile] [-io inoutfile] [-c infile_l1c] \
  [-o outfile_bufr] [-m mwsfile]
```

Notes:

- If the "-io" option is used then the software will attempt to create a "derived" group in the input netCDF file (containing datasets such as *derived/mws_mapped_bt*, *derived/reconstructed_radiances_b1*, etc.). If you want to avoid modifying the input file then create a copy before running the executable. The "-i" and "-io" options cannot be used simultaneously.
- If the IASI-NG input is BUFR, then you should use the "-c" option if the input file is level 1c, and the "-i" option if it is level 1d.
- If the inputs are netCDF then the "-i" option would normally be used to specify the input file, but "-c" can be used in addition if you want to use the l1c raw radiances together with l1d PC scores.
- The top-level namelist has the title **NG_main_nl** and has the following options (Table 3):

Table 3: Options in NG_main_nl

Variable	Default	Comment
read_NG_channels	.false.	Is a channel selection required?

convert_NG_to_IASI	.false.	Convert to IASI first generation spectral response?
mws_spatial_filtering	.false.	Carry out spatial filtering for MWS before mapping?
NG_use_raw_radiance	.false.	Use raw instead of reconstructed radiance in level 1d output? If so, requires a l1c file to be present.
compute_pcscores	.false.	Do you want to compute PC scores? If so, the input must be l1c and appropriate ancillary files must be available, as specified in the Ancillary file namelist (Table 5).
channels_namelist	NG_channels_default.nl	Channel selection namelist
ancil_namelist	NG_ancil.nl	Ancillary file namelist
thinning_namelist	NG_thinning.nl	Thinning namelist (see Table 5 and Table 6, and Section 5.3)
bufr_namelist	NG_bufr.nl	BUFR namelist
MWS_averaging_namelist	MWS_averaging.nl	Defines the spatial averaging for MWS (when mws_spatial_filtering is .true.)

The lower-level namelists are defined in Table 4, Table 5, Table 6, Table 7 and Table 8.

Table 4: Channel selection namelist: NG_channels

Variable	**Default**	**Comment**
required_channels_b1	0	List of channels in band 1
required_channels_b2	0	List of channels in band 2 (re-start numbering)
required_channels_b3	0	List of channels in band 3 (re-start numbering)
required_channels_b4	0	List of channels in band 4 (re-start numbering)
required_channels_all	0	List of channels considering all bands together and numbered consecutively
required_channels_iasi	0	List of IASI first generation channels (for use with the ***convert_NG_to_IASI*** option
use_b1	.true.	Use band 1?
use_b2	.true.	Use band 2?
use_b3	.true.	Use band 3?
use_b4	.true.	Use band 4?

Note that the *use_b1 [or 2/3/4]* options can be used to disable a band, if the user knows that it is not needed (perhaps because there has been a band failure).

Table 5: Ancillary file namelist: NG_ancil

Variable	Default	Comment
ancil_dir	" "	Directory where the ancillary files (i.e. the eigenvector files) are to be found
evfile1	" "	Eigenvector file for band 1
evfile2	" "	Eigenvector file for band 2
evfile3	" "	Eigenvector file for band 3
evfile4	" "	Eigenvector file for band 4

Table 6: IASI-NG thinning namelist: thinning_setup

Variable	Default	Comment
do_thinning	.false.	Carry out thinning for IASI-NG?
use_meti_clear	.false.	Choose the field of view with the largest value for *meti_clear_fraction*
warmest_fov_channel	0	Channel in band 1 to use in a "warmest FOV" test
max_latitude_warmest_fov	90.0	Maximum latitude to use in a "warmest FOV" test
thin_nscans	1	Number of scans in a fixed detector sequence (see Table 7)
thin_nfovs	16	Number of FOVs to use in each Field of Regard. Permitted values are 1, 4 or 16, where a value 16 implies full resolution (see Figure 1 and Table 7)

Table 7: IASI-NG thinning namelist: sequence_data

Variable	Default	Comment
detector_sequence	-	An array of dimension (thin_nfovs,thin_nscans) specifying which FOVs are to be use in each scan

For further explanation on the IASI-NG thinning, please see Section 5.3.

Table 8: BUFR namelist: NG_bufr

Variable	Default	Comment
originating_centre	254	WMO centre number (e.g. 254=EUMETSAT)
sub_centre	0	

master_table	45	The IASI-NG BUFR sequences are defined in master table v45
local_table	0	
local_subtype	0	User-defined subtype
write_pcscores	.true.	Include PC scores in level 1d output?
write_channels_or_mws	.false.	Include channel selection or mapped MWS in level 1d output?
use_operator_203	.true.	Use BUFR operator 203 to set the offset for principalComponentScore?

Note: Not all BUFR libraries support the use of BUFR operator 203, which is needed to change the offset for principalComponentScore so that the values fit within the defined range. If *use_operator_203* is set to false then the sequence is modified such that the user can apply the offset change manually after decoding: the offset to be added to the PC scores is held in "nonNormalizedPrincipalComponentScore".

3.3.2 NG_1c_to_1d.sh

NG_1c_to_1d.sh is a script whose function is to convert from level 1c (netCDF) to level 1d (netCDF).

Usage: `NG_1c_to_1d.sh file1c [file1d]`

A typical file name for level 1c is `W_XX-EUMETSAT-Darmstadt,SAT,SGA1-IAS-1C-RAD_C_EUMT_20170616120000_G_D_20070912100359_20070912100630_T_N____.nc`

If no file name is specified for level 1d, then a name will be automatically generated based on the input file but containing the string "1D" instead of "1C".

The generation of PC scores requires a namelist ***NG_ancil.nl*** to be present – defined as in Table 5.

3.3.3 NG_netcdf_to_bufr_1c.exe

The purpose of the *NG_netcdf_to_bufr_1c.exe* executable is to convert level 1c netCDF to level 1c BUFR.

Usage: `NG_netcdf_to_bufr_1c.exe -i infile -o outfile_bufr [-n bufr_namelist]`

By default, the BUFR namelist file is called ***NG_bufr.nl***, and it has the definition shown in Table 8. Note that ***write_pcscores*** and ***write_channels_or_mws*** are ignored as they are not applicable to level 1c.

The input file should normally be netCDF with a suffix ".nc". For test purposes, a BUFR input file (with any suffix other than ".nc") can be accepted – to demonstrate the reversibility of BUFR decoding and encoding.

3.3.4 NG_netcdf_to_bufr_1d.exe

The purpose of the *NG_netcdf_to_bufr_1c.exe* executable is to convert level 1d netCDF to level 1d BUFR.

Usage: `NG_netcdf_to_bufr_1d.exe -i infile -o outfile_bufr [-n bufr_namelist]`

By default, the BUFR namelist file is called ***NG_bufr.nl***, and it has the definition shown in Table 8.

The output file will not contain any reconstructed radiances. If those are required you will need to run ***NG_main.exe*** instead.

The input file should normally be netCDF with a suffix ".nc". For test purposes, a BUFR input file (with any suffix other than ".nc") can be accepted – to demonstrate the reversibility of BUFR decoding and encoding.

3.4 MWS processing

The top-level routines for MWS processing are listed in Table 9.

Table 9: Main program for MWS

Executable	Functionality
mws_main.exe	• Read input file in netCDF or BUFR • Write output as BUFR, either "pure" level 1b at full spatial resolution or at reduced resolution and optionally including derived fields. • Perform spatial filtering, to reduce noise • Perform scattering and surface tests (see RD-4 and RD-5) • If the input file is netCDF, there is the option to append information in a "derived" group
mws_netcdf_to_bufr_1b.exe	• Read input file in netCDF (or BUFR, for test purposes) • Write output as BUFR level 1b

3.4.1 mws_main.exe

Usage: `mws_main.exe [-n namelistfile] [-i infile] [-io inoutfile_nc] [-o outfile_bufr]`

Notes:

- The input file (level 1b) can be BUFR or netCDF.
- If the "-io" option is used then the software will attempt to create a "derived" group in the input netCDF file (e.g. containing dataset *derived/mws_filtered_bt*). If you want to avoid modifying the input file then create a copy before running the executable. The "-i" and "-io" options cannot be used simultaneously.

- The top-level namelist has the title **mws_main_nl** and has the following definitions (Table 10):

Table 10: Options in mws_main_nl

Variable	Default	Comment
MWS_averaging_namelist	MWS_averaging.nl	Defines the spatial averaging
MWS_microwave_namelist	MWS_microwave_tests.nl	Constants for the scattering and surface tests
MWS_bufr_namelist	MWS_bufr.nl	BUFR namelist

The lower-level namelists are defined in Table 11, Table 12 and Table 13.

Table 11: MWS averaging namelist: MWS_averaging

Variable	Default	Comment
nchan_to_modify	0	Number of channels to modify
channel	-	Array of channel numbers, for the channels that are to be modified
sampling_dist	-	Sampling distance, typically in degrees (single value applicable to all channels)
beam_width	-1.0	Array of beam widths
new_width	-1.0	Array of target beam widths
cutoff	-1.0	Array of cutoff values
nxaverage	-1	Along-scan averaging for the channels that are to be modified
nyaverage	-1	Cross-scan averaging for the channels that are to be modified
qc_dist	-1	Quality threshold (in units of the sampling distance) for the channels that are to be modified. Scans or spots that cannot be accurately filtered because of proximity to a gap are flagged.

The spatial filtering procedure is based on that implemented in AAPP for ATMS. This is described in more detail in Section 5.4.

Table 12: MWS pre-processing tests: mws_prepro_tests

Variable	Default	Comment
ASecAng	-	Secant of 5 zenith angles
coefs_scat	-	89GHz predicted from 23.8, 31.4, 50.3: 4 coefs each for constant term and 3 channels

coefs_cirr	-	183+-7GHz predicted from 23.8, 89, 166: 4 coefs each for constant term and 3 channels
coefs_cirr2	-	229GHz predicted from 89, 166, 183+-7): 4 coefs each for constant term and 3 channels
ASrfChn	-	Channels used in surface test
AmeanBT	-	Mean BT for 3 channels, 5 angles, 6 surface types
Acov	-	Covariance for 3 channels, 3 channels, 5 angles, 6 surface types

Table 13: MWS BUFR options: mws_bufr

Variable	**Default**	**Comment**
originating_centre	255	255=missing
sub_centre	0	
master_table	40	
local_table	0	
local_subtype	0	User-defined subtype
scans_per_message	1	Number of MWS scans per BUFR message
spot_thin	1	Output 1 every *N* spots
scan_thin	1	Output 1 every *N* scans
append_scattering_indices	.false.	Include scattering indices and surface type using local descriptors

3.4.2 mws_netcdf_to_bufr_1b.exe

Usage: `mws_netcdf_to_bufr_1b.exe [-n bufr_namelistfile] [-i infile] [-o outfile_bufr]`

This is a simple program to convert netCDF to level 1b BUFR (e.g. for use at DBNet stations). The options for the BUFR namelist are the same as in Table 13.

3.5 Display utilities

A python tool AAPP_BUFR_quicklook.py has been provided to give a quick-look display of the BUFR-encoded MWS brightness temperatures or IASI-NG radiances:

This uses python3.x and requires the following python libraries to be available: numpy, matplotlib, cartopy[5] and eccodes[6].

[5] https://cartopy.readthedocs.io/stable/
[6] https://confluence.ecmwf.int/display/ECC

It should work for any BUFR file that contains the following descriptors: *latitude*, *longitude* and either *brightnessTemperature* or *channelRadiance*. The product is displayed on a regular lat/lon (Plate Carrée) grid.

For usage instructions, use the argument "-h".

You may need to set up your ECCODES_DEFINITION_PATH before running the tool, e.g.

```
export ECCODES_DEFINITION_PATH=$AAPP_SG_INSTALL/share
```

4. TEST CASES

As shown in the previous sections, there are many possible permutations of input data, output data and processing options. We have provided test cases for typical combinations that the user may wish to use, available for download from https://nwp-saf.eumetsat.int/site/software/aapp-sg/ . These are listed in Table 14. The user is encouraged to customise the test cases to suit their own requirements.

You can find the README.md and run_all.sh script in the top directory of the untarred test cases directory. Please refer to the README.md for further detail. Note that the `run_all.sh` script provides an easy way to run all, or a subset, of the tests listed below.

The README.md can be viewed in any terminal or GUI editor. To view the fully formatted file, you may wish to use *Visual Studio Code* as follows: type `code README.md` and press CTRL + SHIFT + V (or right-click on file and select "Open Preview") to turn on formatting.

Table 14: AAPP-SG test cases

No.	Test case	Top-level namelist (where applicable)	Comment
1	MWS_netcdf_1b_to_bufr_1b	MWS_bufr_1b.nl	MWS conversion from level 1b netcdf to a standard level 1b BUFR at full spatial resolution.
2	MWS_netcdf_1b_to_bufr_1d	MWS_main.nl	MWS conversion from level 1b netcdf to a level 1d BUFR file. The MWS BUFR output is spatially thinned and channel BTs have been spatially averaged.
3	MWS_bufr_1b_to_bufr_1d	MWS_main.nl	As above, but for 1b BUFR input.
4	IASI_NG_netcdf_1c_to_bufr_1d_IASI_SRF	NG_main_1c_to_1d.nl	IASI-NG conversion from level 1c netCDF to level 1d BUFR at IASI first generation spectral resolution. Outputs come from the raw radiances.
5	IASI_NG_netcdf_1c_to_bufr_1c	NG_bufr_1c.nl	IASI-NG conversion from level 1c netCDF to a standard level

			1c BUFR at full spatial resolution.
6	IASI_NG_netcdf_1d_to_bufr_1d	NG_main_1d.nl	IASI-NG conversion from level 1d netCDF (PC scores) to a standard level 1d BUFR at full spatial resolution.
7	IASI_NG_netcdf_1d_to_bufr_1d_IASI_SRF	NG_main_iasi314.nl	IASI-NG processing from level 1d netCDF (PC scores) to “enhanced” level 1d BUFR, including spatial thinning and conversion to IASI first generation spectral resolution. Outputs are also fed back to the input netCDF file.
8	IASI_NG_netcdf_1d_to_bufr_1d_with_MWS	NG_main_with_mws.nl	IASI-NG processing from level 1d netCDF (PC scores) to “enhanced” level 1d BUFR, including mapped MWS.
9	IASI_NG_1c_to_1d_netcdf	-	IASI-NG conversion from level 1c netCDF (full spectrum) to level 1d netCDF (PC scores). This can be used in as a follow-on step to the Direct Broadcast package which generates level 1c as its output.
10	IASI_NG_bufr_1c_to_bufr_1d	NG_main_1c_to_1d.nl	IASI-NG conversion from level 1c BUFR to level 1d BUFR at IASI first generation spectral resolution. Outputs come from the raw radiances.
11	IASI_NG_bufr_1d_to_bufr_1d	NG_main_1d_to_1d.nl	As for test 10, but for level 1d BUFR input. Outputs are reconstructed radiances.
12	IASI_NG_bufr_1d_to_bufr_1d_with_MWS	NG_main_with_mws.nl	As for test 11, but the inputs are BUFR.

5. SCIENTIFIC DESCRIPTION

This section describes science issues relevant to AAPP-SG. For general information on the EPS-SG instruments, please see https://www.eumetsat.int/metop-sg.

5.1 IASI-NG Principal Components manipulations

As stated in [RD-7], Principal Components (PC) scores, **p**, are related to the measured radiance spectrum, **r** (column vector), and the reconstructed radiance spectrum, **r**’, as follows:

$$\mathbf{p} = \mathbf{E}^T \mathbf{N}^{-1} (\mathbf{r} - \mathbf{r}_m) \tag{1}$$

$$\mathbf{r}' = \mathbf{r}_m + \mathbf{N}\,\mathbf{E}\,\mathbf{p} \tag{2}$$

where $\mathbf{r_m}$ is a fixed mean radiance spectrum, **N** is the noise normalisation matrix (usually the matrix square root of the instrument noise covariance) and **E** is a truncated set of eigenvectors ($\mathbf{E}^T \mathbf{E} = \mathbf{I}$, where **I** is the identity matrix). Note that **E** is of rank {*nchan* rows, *npc* columns}.

To minimise the number of matrix multiplications being carried out in near-real-time, a compression operator $\mathbf{C} = \mathbf{E}^T \mathbf{N}^{-1}$ and a reconstruction operator $\mathbf{R} = \mathbf{N}\,\mathbf{E}$ are defined, so that:

$$\mathbf{p} = \mathbf{C}\,(\mathbf{r} - \mathbf{r_m}) \qquad (3)$$

$$\mathbf{r'} = \mathbf{r_m} + \mathbf{R}\,\mathbf{p} \qquad (4)$$

AAPP-SG has the capability to convert raw radiances to PC scores using Equation (3), and also to convert PC scores to reconstructed radiance using Equation (4). It is the latter process that is more likely to be of interest to operational users.

The necessary compression operators, reconstruction operators and mean radiances will be supplied by EUMETSAT, and specified in the AAPP-SG Auxiliary file namelist. The process is done separately for each IASI-NG band – where there are 4 bands as specified in Table 15.

Table 15: IASI-NG spectral band definitions

Band	Number of channels	Start channel	End channel	Start wavenumber (cm^{-1})	End wavenumber (cm^{-1})
1	4040	1	4040	645	1149.875
2	6400	4041	10440	1150	1949.875
3	2800	10441	13240	1950	2299.875
4	3681	13241	16921	2300	2760

5.2 Conversion to IASI first generation spectral response

Some users will wish to convert IASI-NG data to the spectral response function of IASI first generation – especially for "day-1" use in assimilation systems that already process IASI first generation.

This can be done in two ways:

1. From the level 1c spectra – by changing the apodisation of each spectrum
2. From the level 1d PC scores – by changing the apodisation of the reconstruction operators.

The second method will be faster computationally as the modified reconstruction operators can be used for all spectra in the input file. However, the theory is basically the same for the two methods.

Both IASI-NG and IASI have gaussian apodisation functions in the interferogram domain. This results in approximately gaussian spectral response functions – apart from a small amount of ringing due to the fact that the interferogram is truncated at maximum Optical Path Difference (OPD).

The ideal response in the spectral domain (assuming no OPD truncation) is given by

$$h(\nu) = e^{-\ln(2)\left(\frac{\nu}{\nu_0}\right)^2} \qquad (5)$$

where ν_0 is 0.25 cm^{-1} for IASI and 0.125 cm^{-1} for IASI-NG. The response is 0.5 when $\nu = \nu_0$.

The corresponding response in the interferogram domain is

$$H(x) = e^{-\ln(2)\left(\frac{x}{x_0}\right)^2} \qquad (6)$$

where x is optical path difference and $x_0 = \ln(2) / (\pi\, \nu_0) =$ 0.8825424 cm for IASI or 1.7650848 cm for IASI-NG. Note that IASI has a maximum OPD of 2.0 cm and IASI-NG has a maximum OPD of 4.0 cm.

Thus, the ratio of the responses for the 2 instruments in interferogram domain is

$$\frac{H(x)_{IASI}}{H(x)_{IASING}} = e^{-\ln(2)\left\{\left(\frac{x}{x_{0\_IASI}}\right)^2 - \left(\frac{x}{x_{0\_IASING}}\right)^2\right\}}$$

$$= e^{-0.75\ln(2)\left(\frac{x}{x_{0\_IASI}}\right)^2} \qquad (7)$$

To generate the interferogram response from the spectrum using a Fourier Transform, we need to be careful to avoid edge effects, which could generate a large amount of ringing. Therefore the following approach is taken in the software, for each spectrum or eigenvector:

1. Pad the spectrum or eigenvector by appending a partial mirror image at both ends, so that the number of points is a power of 2. For example, if the input spectrum has 16921 points, then pad it to 32768 points (i.e. 2^{15}).
2. Duplicate the spectrum by appending its mirror image – i.e. if the padded spectrum has 32768 points, then we get a total of 65536 points, with no discontinuities on wrap-around.
3. Fourier transform to interferogram domain
4. Multiply by the exponential function in Equation (7)
5. Inverse Fourier transform back to spectral domain
6. Discard the padding

5.3 IASI-NG spatial thinning

IASI-NG has 16 fields of view (FOV) per field of regard (FOR) and there are 14 FORs in a scan line – giving a total of 224 FOVs per scan.

In the thinning routine, a mask is created which defines which samples are to be output. It supports either 4 FOVs per FOR (i.e. similar to AAPP thinning option in IASI first generation) or 1 FOV per -FOR.

Several methods are available, selectable via namelist, for choosing which FOVs to select:

1. Use the FOV with the highest METimage clear fraction (if this parameter varies within the FOR).
2. A fixed set of FOVs in each field of regard. The user may choose a different set for different scan e.g. a 4-scan repeating pattern. The FOVs are ordered according to Figure 1.
3. Warmest IASI-NG FOV for a specified channel in band 1. If using this option, the array NG_recon_rad_b1 must have been populated, either with reconstructed or raw radiances.
4. Method 2 in polar regions (defined by a latitude threshold) and method 3 elsewhere.

Options 2 to 4 mirror those available in AAPP-core for CrIS. Option 1 is new and is to be evaluated post-launch.

The principle behind option 4 is that in polar regions, with ice cover, it is often found that the warmest FOV is not necessarily the one most likely to be cloud-free. Therefore, a fixed detector sequence may perform better in this situation.

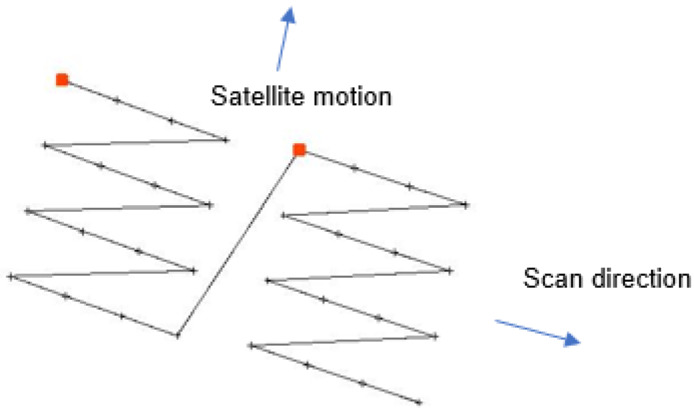

Figure 1: Ordering of IASI-NG FOVs. The diagram shows two consecutive fields of regard, each containing 16 FOVs. FOV number 1 is marked with a red square.

5.4 MWS spatial filtering

For users who wish to spatially average MWS data in order to reduce noise, and possibly equalise the channel footprints, two methods are provided in AAPP-SG:

- A simple averaging of *N* spots cross-track and *M* scans along-track
- FFT-based technique as described in section 3 of [RD-6]

To understand how much averaging may be needed, it is helpful to consider the channel characteristics. See Table 16. (The values are consistent with the MWS Science Plan[7] and an altitude of 827 km).

[7] MWS Science Plan: https://www.eumetsat.int/media/43204

Table 16: MWS channel beam widths

Channels	Frequencies (GHz)	Cross-track sampling distance (degrees)	Beam width (degrees)
1-2	23.8, 31.4	1.04919	2.74
3-16	50.3 to 57.29	1.04919	1.39
17-24	89 to 229	1.04919	1.18

The scan time is 2.254 seconds, so the along-track sampling distance is 14.89 km, which is equivalent to an angle of 1.03 degrees.

If we average 3 samples cross-track and 3 samples along-track, we get an effective beam width at nadir for all channels of *approximately* 3.15°, which is close to that of AMSU-A (3.33°). Of course, in practice the beam shapes differ for the different channel groups; see Figure 2. But they are closer in width after averaging compared with before averaging.

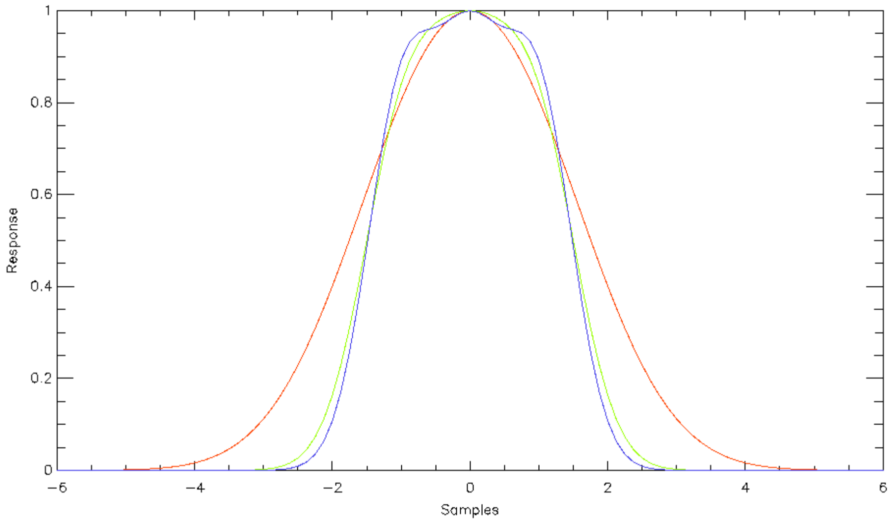

Figure 2: MWS beam shapes when 3 successive IFOVs are averaged. The result is normalised to 1. Red: channels 1-2, Green: channels 3-16, Blue: channels 17-24. The finite integration time has been ignored.

It would be expected that the FFT-based method (as used for ATMS) could give greater consistency between the channel groups – if that is required. But for NWP purposes we suspect there is no strong benefit to having channels 1 and 2 exactly the same width as the sounding channels. This does not seem to be a great concern in ATMS experience.

5.5 MWS scattering and surface tests

The scattering and surface tests are described in detail in RD-4 and RD-5. They are summarised in this section.

5.5.1 Scattering tests

To predict the BT of a channel *m*, from channels *i*, *j*, *k*, AAPP-SG uses the equation

$$\widetilde{T_m} = \left(1, T_i, T_j, T_k\right) \mathbf{M} \begin{pmatrix} 1 \\ x \\ x^2 \\ x^3 \end{pmatrix} \tag{8}$$

Where the tilde indicates a prediction, x is 1-sec(z) where z is the zenith angle of the satellite as viewed from earth, and **M** is a 4x4 matrix of coefficients – the columns provide zenith angle dependence, and the rows provide a linear combination of the channels (the first row is a constant term).

The scattering index is then $\widetilde{T_m} - T_m$ (normally positive because scattering depresses the observed BTs).

The method of deriving the coefficients is described in RD-4.

The following scattering indices are generated in AAPP-SG:

1. Scattering test (ocean): predict 89 GHz from 23.8, 31.4 and 50.3 GHz
2. Legacy cirrus test (ocean): predict 183±7 GHz from 23.8, 89 and 166 GHz
3. New cirrus test (all surfaces): predict 229 GHz from 89, 166 and 183±7 GHz

In addition, a “Bennartz rain test” is run using 23.8 (land only), 89 and 166 GHz. This test was originally developed for AMSU/MHS by the NWC SAF. Over ocean, the scattering index is defined as:

$$\mathrm{SI} = T_{89\ GHz} - T_{150\ GHz} - \mathrm{offset(K)} \tag{9}$$

where offset = a + b SatAng. A rain flag is triggered when SI > threshold. See RD-4 for an outline of the Bennartz rain test over land.

Only the scattering index is written to AAPP-SG output BUFR files, not the rain flags.

5.5.2 Surface test

The surface test is described in RD-5. It works by looking for the surface type that minimises a cost function J based on the brightness temperatures of MWS channels 1, 2 and 3 (23.8, 31.4 and 50.3 GHz):

$$J = \left(T_1 - \overline{T_1}\ ,\quad T_2 - \overline{T_2}\ ,\quad T_3 - \overline{T_3}\right) \mathbf{C}^{-1} \begin{pmatrix} T_1 - \overline{T_1} \\ T_2 - \overline{T_2} \\ T_3 - \overline{T_3} \end{pmatrix} \tag{10}$$

where **C** is a 3x3 covariance matrix. Values of **C**, $\overline{T_1}$, $\overline{T_2}$ and $\overline{T_3}$ are provided for 5 different fixed zenith angles (sec(z) = 1.0,1.25,1.5,1.75,2.0), and they are interpolated for each observation.

Six surface types are defined, corresponding to different combinations of the surface types in the TELSEM atlas. See Table 17.

Table 17: Translation of TELSEM categories to six categories

category number	TELSEM Class2	Characteristic
1	None, 11, 12, 13	Ocean and new sea-ice
2	20, 21, 22, 14, 15, 16	Multi-year ice
3	1, 2, 10	Wet land and forest
4	3, 4	Dry land
5	17, 18, 19	Snow
6	5	Desert

The most probable surface type, and the value of the cost function are written to the output MWS level 1d BUFR file.

High values of the cost function can be used to indicate the presence of clouds that obscure the surface and make precise determination of the surface type impossible.

6. BUFR SEQUENCES

6.1 IASI-NG level 1C

The IASI-NG BUFR sequences are not yet finalised. WMO sequences will be made available soon (replacing some provisional sequences), but for now we write the descriptors explicitly based on information received from EUMETSAT in late 2025. The sequence is in two parts, a) and b):

a) common sequence for (i) IASI-NG date/time/geolocation/flags, (ii) METimage quantities

```
! 001007 satelliteIdentifier
! 001035 centre, note [001035,001034] = 301005
! 001034 subCentre
! 002019 #1#satelliteInstruments (231=IASI-NG)
! 002020 satelliteClassification (62=EPS-SG)
! 004001 year
! 004002 month
! 004003 day
! 004004 hour
! 004005 minute
! 202131
! 201138
! 004006 second
! 201000
! 202000
! 005001 latitude
! 006001 longitude
! 007026 satelliteZenithAngle
! 202130
```

```
! 201135
! 005021 bearingOrAzimuth
! 201000
! 202000
! 007025 solarZenithAngle
! 005022 solarAzimuth
! 005043 fieldOfViewNumber (1 to 224)
! 005040 orbitNumber
! 201133
! 005041 scanLineNumber
! 201000
! 202126
! 007001 heightOfStation
! 202000
! 033113 iasiNgGeneralQualityFlags
! 002019   #2#satelliteInstruments (CCT-C8, 232=METimage)
! 115000   [loop over classes]
! 031002   extendedDelayedDescriptorReplicationFactor
!   005060   yAngularPositionFromCentreOfGravity
!   005061   zAngularPositionFromCentreOfGravity
!   202129
!   201132
!   020083   amountOfSegmentCoveredByScene
!   201000
!   202000
!   106000   [loop over channels]
!   031002   extendedDelayedDescriptorReplicationFactor
!   005042     channelNumber
!   008023     firstOrderStatistics (4=mean)
!   014043     channelRadiance (W m-2 sr-1 um-1)
!   008023     firstOrderStatistics (10=std)
!   014043     channelRadiance (W m-2 sr-1 um-1)
!   008023     firstOrderStatistics (missing)
! 008072    pixelType (1=clear, meti_clear_fraction)
! 202129
! 201132
! 020083    amountOfSegmentCoveredByScene
! 201000
! 202000
! 008072    pixelType (3=prob_clear, meti_prob_clear_fraction)
! 202129
! 201132
! 020083    amountOfSegmentCoveredByScene
! 201000
! 202000
! 008072    pixelType (4=prob_cloudy, meti_prob_cloudy_fraction)
! 202129
! 201132
! 020083    amountOfSegmentCoveredByScene
! 201000
! 202000
! 008072    pixelType (2=cloudy, meti_cloudy_fraction)
! 202129
! 201132
! 020083    amountOfSegmentCoveredByScene
! 201000
! 202000
! 008072    pixelType (5=ash, meti_ash_fraction)
! 202129
```

```
! 201132
! 020083    amountOfSegmentCoveredByScene
! 201000
! 202000
! 008072    pixelType (missing)
! 008029    surfaceType (13=sea, meti_sea_fraction)
! 202129
! 201132
! 020083    amountOfSegmentCoveredByScene
! 201000
! 202000
! 008029    surfaceType (19=forest, meti_forests_fraction)
! 202129
! 201132
! 020083    amountOfSegmentCoveredByScene
! 201000
! 202000
! 008029    surfaceType (20=shrublands, meti_shrublands_fraction)
! 202129
! 201132
! 020083    amountOfSegmentCoveredByScene
! 201000
! 202000
! 008029    surfaceType (21=savanna, meti_savannas_fraction)
! 202129
! 201132
! 020083    amountOfSegmentCoveredByScene
! 201000
! 202000
! 008029    surfaceType (22=desert, meti_desert_fraction)
! 202129
! 201132
! 020083    amountOfSegmentCoveredByScene
! 201000
! 202000
! 008029    surfaceType (23=grasslands, meti_grasslands_fraction)
! 202129
! 201132
! 020083    amountOfSegmentCoveredByScene
! 201000
! 202000
! 008029    surfaceType (24=wetlands, meti_wetlands_fraction)
! 202129
! 201132
! 020083    amountOfSegmentCoveredByScene
! 201000
! 202000
! 008029    surfaceType (15=urban land, meti_urban_fraction)
! 202129
! 201132
! 020083    amountOfSegmentCoveredByScene
! 201000
! 202000
! 008029    surfaceType (16=rural land, meti_croplands_fraction)
! 202129
! 201132
! 020083    amountOfSegmentCoveredByScene
! 201000
! 202000
```

```
! 008029    surfaceType (25=snow/ice, meti_snow_ice_fraction)
! 202129
! 201132
! 020083    amountOfSegmentCoveredByScene
! 201000
! 202000
! 008029    surfaceType (26=coast, meti_coast_fraction)
! 202129
! 201132
! 020083    amountOfSegmentCoveredByScene
! 201000
! 202000
! 008029    surfaceType (missing)
```

b) IASI-NG radiances

```
! 002019 #3#satelliteInstruments (231=IASI-NG)
! 105170 [replicate *170]
!   104100 [replicate *100]
!     201137 [change width]
!     005042 channelNumber
!     201000 [reset width]
!     014044 channelRadiance (W m-2 sr-1 cm)
```

Note that the ecCodes mnemonic *channelRadiance* is used both for METimage and for IASI-NG – but with different units, reflecting those present in the original netCDF (W m^{-2} sr^{-1} um^{-1} for METimage and W m^{-2} sr^{-1} cm for IASI-NG).

6.2 IASI-NG level 1D

This sequence contains three parts. The BUFR product to be distributed by EUMETSAT will comprise parts a) and b). Part c) is provided as an optional addition in AAPP-SG, for users who require mapped MWS or an IASI-NG channel selection (raw radiances or reconstructed, depending on the input).

a) common sequence for (i) IASI-NG date/time/geolocation/flags, (ii) METimage quantities. See section 6.1.

b) IASI-NG PC scores. Note that the *principalComponentScore* is computed from noise-normalised radiances and has precision 0.1 in this sequence.

As mentioned in section 6.1, this sequence may change. Note that it makes use of the 203xxx operator to change a reference value, a facility that was only introduced into ecCodes in version 2.44.0 (released October 2025) even though it has been part of the BUFR standard for many years. To allow decoding by systems that do not support the 203 operator in the same way as ecCodes (e.g. the Met Office BUFR library), we have implemented an option, controlled by namelist (see Table 8), in which “203021 025050 203255” is replaced by “202129 040017 202000”.

```
! 002019 #3#satelliteInstruments  (CCT-C8, 231=IASI-NG)
! 203021 [change reference value using 21 bits]
! 025050 inputOverriddenReferenceValue (for principalComponentScore, set to -524288)
! 203255 [reset reference value]
! 116004   [replication over 4 bands]
!   201129  [change width]
```

```
!   025140  startChannel
!   025141  endChannel
!   201000  [reset width]
!   202134  [change scale]
!   201148  [change width]
!   040016  residualRmsInBand
!   201000  [reset width]
!   202000  [reset scale]
!   105000  [delayed replication of 5 descriptors]
!   031002  [number of PC scores in band]
!   202125  [change scale]
!   201130  [change width]
!   025050  principalComponentScore
!   201000  [reset width]
!   202000  [reset scale]
```

c) Optional: IASI-NG channel selection and mapped MWS

```
! 002165 radianceTypeFlags (4=apodised, 6=reconstructed)
! 104000 [delayed replication of 4 descriptors]
! 031002 [number of channels (extendedDelayedDescriptorReplicationFactor)]
!   201137   [change width]
!   005042   channelNumber
!   201000   [reset scale]
!   014044   channelRadiance (W m-2 sr-1 cm)
! 002019 #4#satelliteInstruments (CCT-C8, 233=MWS)
! 104000 [delayed replication of 4 descriptors]
! 031001 [number of channels (delayedDescriptorReplicationFactor)]
!   005042   channelNumber
!   207002   [change scale and width]
!   012063   brightnessTemperature (to 0.001K)
!   207000   [reset scale and width]
```

6.3 MWS level 1B

The following sequence was first defined in BUFR tables v40 (June 2023[8])

```
! 310082 [MWS L1B] is defined as
!
!   008070 verticalSoundingProductQualifier (2=level 1B)
!   001033 centre (CCT-C1)
!   001034 subCentre (CCT-C12)
!   001007 satelliteIdentifier (CCT-C5, 24=Metop-D)
!   002019 satelliteInstruments (CCT-C8, 233=MWS)
!   005040 orbitNumber
!   025061 softwareVersionNumber (string, e.g. "v1_Global")
!   201133,005041,201000 scanLineNumber
!   005043 fieldOfViewNumber
!   006042 spaceOrMoonScanSampleNumber
!   301011 year, month, day
!   301012 hour, minute
!   207003,004006,207000 second (0.001s precision)
!   301021 latitude,longitude (0.00001 deg precision)
!   107002
!     005080 resolutionRadius
```

[8] https://community.wmo.int/en/activity-areas/wis/publications/306-vI.2/previous-versions

```
!     201132,202129,021166,202000,201000 landFraction
!     007007 height
!   202126,007001,202000 satellite height (100m precision)
!   007024 satelliteZenithAngle (0.01 deg precision)
!   005021 bearingOrAzimuth
!   007025 solarZenithAngle
!   005022 solarAzimuth
!   201124,202127,025084,202000, 201000 geometricRotationalAngle (orbit angle, precision
reduced to 0.001 deg)
!   007073 angleBetweenMoonAndSpaceView
!   012070 warmLoadTemperature
!   012064 instrumentTemperature
!   008098 sourceOfTemperatureMeasurement (e.g. 5=rotating reflector)
!   012101 airTemperature
!   008098 sourceOfTemperatureMeasurement (set to missing)
!   033107 mwsScanLineProcessingFlags
!   033108 mwsNavigationStatusFlags
!   033109 mwsOverallQualityFlags
!   116024 (replicate 16 descriptors 24 times)
!     005042 channelNumber
!     102004
!       005079 bandNumber
!       002154 satelliteChannelBandWidth (better: 202130,002154,202000)
!     002104 antennaPolarization
!     002153 satelliteChannelCentreFrequency
!     025077 bandwidthCorrectionCoefficient1
!     025078 bandwidthCorrectionCoefficient2
!     207002,012063,207000 brightnessTemperature
!     207001 (scale change)
!       033089 noiseEquivalentDeltaTemperatureNedtQualityIndicatorsForWarmTargetCalibration
!       033090 nedtQualityIndicatorsForColdTargetCalibration
!     207000
!     033110 mwsChannelProcessingFlags
!
! Optional additions (if "append_scattering_indices" is true):
!
!    008205 estimatedNearestAmsuSurfaceType
!    002224 surfaceCostFunction
!    025206 scatteringIndex (89 GHz, sea only)
!    025206 scatteringIndex (NWCSAF test, 166 GHz, all surfaces)
!    025208 microwaveCirrusIndex (183 GHz, sea only)
!    025208 microwaveCirrusIndex (229 GHz, all surfaces)
```

Please note that the “optional additions” rely on local descriptors for centre 74 (Met Office). These are included under “definitions” in AAPP-SG.