GBGP User Guide

Version 1.0

31 March 2019

Danish Meteorological Institute (DMI)
European Centre for Medium-Range Weather Forecasts (ECMWF)
Institut d'Estudis Espacials de Catalunya (IEEC)
Met Office (METO)

Ref: SAF/ROM/METO/UG/GBGP/001

DOCUMENT AUTHOR TABLE

	Author(s)	*Function*	*Date*	*Comment*
Prepared by:	D Offiler O Lewis	GBGP Development Team Leader GBGP Development Team	31/03/2019	
Reviewed by:	I Culverwell	ROM SAF Package Manager	31/03/2019	
Approved by:	K B Lauritsen	ROM SAF Project manager	31/03/2019	

DOCUMENT CHANGE RECORD

Issue/Revision	*Date*	*By*	*Description*
1.0	31/03/19	DO/OL	1st Release version GBGP-1 (V1.0)

ROM SAF

The Radio Occultation Meteorology Satellite Application Facility (ROM SAF) is a decentralised processing centre under EUMETSAT which is responsible for operational processing of GRAS radio occultation data from the Metop satellites and radio occultation (RO) data from other missions. The ROM SAF delivers bending angle, refractivity, temperature, pressure, and humidity profiles in near-real time and off-line for NWP and climate users. The off-line profiles are further processed into climate products consisting of gridded monthly zonal means of bending angle, refractivity, temperature, humidity, and geopotential heights together with error descriptions.

The ROM SAF also maintains the Radio Occultation Processing Package (ROPP) which contains software modules that will aid users wishing to process, quality-control and assimilate radio occultation data from any radio occultation mission into NWP and other models, and the Ground-Based GNSS Package (GBGP) which provides format conversion with quality-checking for processed ground-based GNSS data prior to dissemination to, and use by, NWP centres.

The ROM SAF Leading Entity is the Danish Meteorological Institute (DMI), with Cooperating Entities: i) European Centre for Medium-Range Weather Forecasts (ECMWF) in Reading, United Kingdom, ii) Institut D'Estudis Espacials de Catalunya (IEEC) in Barcelona, Spain, and iii) Met Office in Exeter, United Kingdom. To get access to our products or to read more about the ROM SAF please go to: http://www.romsaf.org

Intellectual Property Rights

Contents

Index of Tables

Index of Illustrations

Executive Summary

This document outlines the first full release of the Ground-Based GNSS Package (GBGP) Version 1.0 or just **GBGP-1**. It describes the GBGP structure and the use of the principle library functions, executable tools, supporting scripts and run-time data files provided in the package.

GBGP is a collection of software (provided as source code), supporting build scripts, data files and documentation. Its principle purpose is to encode E-GVAP 'COST-format' files to WMO-standard BUFR format for dissemination to NWP users locally or globally via the GTS. The package also contains several pre-conversion tools to convert other formats, such as netCDF and CSV, to COST-format prior to BUFR encoding. Support for quality control against an SQL database of GNSS station meta-data is also included. Users may wish to integrate a subset of GBGP code into their own software applications, for instance linking the GBGP library to their own code and/or adapting one of the conversion tools to input data a different format.

For details on the files in the package, required 3rd-party dependency packages and guidance on building and installing the GBGP software please refer to the GBGP Release Notes [RD.2].

This release is designated 'full' or ‘operational’ as it has undergone the ROM SAF's formal release procedure which includes code review, a comprehensive test suite, independent (external) beta-testing, a Delivery Readiness Inspection Review and formal approval to release by the ROM SAF Steering Group as fully able to meet operational requirements.

1. Introduction

1.1 Purpose of the document

This document outlines the first full release of the Ground-Based GNSS Package (GBGP) Version 1.0 or just **GBGP-1**. It describes GBGP structure and the use of the principle library functions, executable tools, supporting scripts and run-time data files provided in the package.

1.2 Applicable and reference documents

1.2.1 Applicable documents

The following list contains documents with a direct bearing on the contents of this document:

[AD.1] CDOP-2 Proposal: Proposal for the Second Continuous Development and Operations Phase (CDOP-2); Ref: SAF/GRAS/DMI/MGT/CDOP2/001 Version 1.1 of 21 March 2011, approved by the EUMETSAT Council in Ref. EUM/C/72/11/DOC/10 at its 72nd meeting on 28-29 June 2011;

[AD.2] CDOP-2 Cooperation Agreement: Agreement between EUMETSAT and DMI on the Second Continuous Development and Operations Phase (CDOP-2) of the Radio Occultation Meteorology Satellite Applications Facility (ROM SAF), approved by the EUMETSAT Council; Ref: EUM/C/72/11/DOC/15 at its 72nd meeting on 28-29 June 2011 and signed on 29 June 2011 in Copenhagen;

1.2.2 Reference documents

The following documents provide supplementary or background information, and could be helpful in conjunction with this document:

[RD.1] GBGP Change Log
Ref: SAF/ROM/METO/CL/GBGP/001

[RD.2] GBGP Release Notes
Ref: SAF/ROM/METO/RN/GBGP/001

[RD.3] GBGP Reference Manual
Ref: SAF/ROM/METO/RM/GBGP/001

[RD.4] 'COST-format' file specification for ground-based GNSS delay and water vapour data
Ref: E-GVAP/METO/FMT/COST/001

[RD.5] WMO FM94 (BUFR) specification for ground-based GNSS delay and water vapour data
Ref: E-GVAP/METO/FMT/BUFR/001

[RD.6] NetCDF (Unidata) website
URL: http://www.unidata.ucar.edu/software/netcdf/

[RD.7] NOAA/Earth Systems Research Laboratory GPSNet website
URL: http://gpsmet.noaa.gov/

[RD.8] UCAR/COSMIC Programme Office SuomiNet/CONUS website
URL: http://www.suominet.ucar.edu/

[RD.9] SQLite database website
URL: http://www.sqlite.org/

[RD.10] SQLite Manager website
URL: http://lazierthanthou.github.io/sqlite-manager/

[RD.11] SQLite Browser website
URL: http://sqlitebrowser.org/

[RD.12] ISO-3166 English country names and their 2-letter country codes are listed at:
URL:
http://www.iso.org/iso/country_codes/iso_3166_code_lists/english_country_names_and_code_elements.htm
See also: http://en.wikipedia.org/wiki/ISO_3166-1_alpha-2

[RD.13] WMO OSCAR/Surface database:
https://oscar.wmo.int/surface/index.html
The Vol.A legacy file[1] can be downloaded from:
http://oscar.wmo.int/oscar/vola/vola_legacy_report.txt

[RD.14] CGI browser tool[2]
Internet URL: http://daveoffiler.uk/cgi-bin/gbgnss.cgi
MetO CDN URL: http://www-nwp/~gbgnssman/gbgnss.cgi

1.3 Acronyms and abbreviations

AC	Analysis (Processing) Centre
AIX	Advanced Interactive eXecutive (IBM)
AMSL	Above Mean Sea Level
API	Application Programming Interface
ARH	Abbreviated Routing Header (GTS)
BUFR	Binary Universal Form for the Representation of data (also: FM94) (WMO)
CDOP-2	Second Continuous Development and Operations Phase (EUMETSAT)
CGI	Computer Gateway Interface
CONUS	Contiguous United States
COSMIC	Constellation Observing System for Meteorology, Ionosphere and Climate
COST	Cooperation in Science and Technology (EU)
CSV	Comma-Separated Value
DOMES	Directory Of MErit Sites[3]
DMI	Danish Meteorological Institute; ROM SAF Leading Entity (Copenhagen, Denmark)
ECMWF	The European Centre for Medium-range Weather Forecasts (Reading, UK)
EGM96	Earth Gravity Model, 1996
E-GVAP	EUMETNET GNSS water VApour Programme
ESRL	Earth Systems Research Laboratory (NOAA, Boulder CO, USA)
EU	European Union
EUMETNET	EUropean METeorological services NETwork
EUMETSAT	EUropean organisation for the exploitation of METeorological SATellites
EUREF	EUropean REFerence frame
EV	Environment Variable
FTP	File Transfer Protocol
GB-GNSS	Ground-Based GNSS
GBGP	Ground-Based GNSS Package
GCC	GNU Compiler Collection (not to be confused with **gcc**, the GCC C-compiler)
GNU	GNU's Not Unix
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GTS	Global Telecommunications System (WMO)
HDF5	Hierarchical Data Format version 5
HPC	High Power Computer (aka 'supercomputer')
ICAO	International Civil Aviation Organization
ID	Identity or Identifier

[1] *This is a 2-year temporary replacement for the old Pub.9 Vol.A 'flatfile' which ceased on 2 May 2016. The legacy file follows the flatfile format though some fields are different.*

[2] *Note: At the time of writing, the Internet-facing URL is a demonstration site only and may be withdrawn during 2017 if/when the tool is hosted on the main E-GVAP website.*

[3] *Project MERIT: Monitor Earth Rotation and Intercompare the Techniques of observation and analysis.*

IEEC	Institut D'Estudis Espacials de Catalunya (Barcelona, Spain)
IGS	International GNSS Service
I/O	Inout/Output
IP	Internet Protocol
IWV	Integrated Water Vapour
Met Office	Meteorological Office of the United Kingdom
MetDB	Meteorological Data Base (Met Office, Exeter, UK)
netCDF	Network Common Data Format
NOAA	National Oceanic and Atmospheric Administration
NRT	Near Real Time
NWP	Numerical Weather Prediction
OS	Operating System
OSGB	Ordnance Survey of Great Britain
OSCAR	Observation Systems Capability Analysis and Review Tool (WMO)
PES	Pre-Existing Software
PNG	Portable Network Graphics
POSIX	Portable Operating System Interface
Q/C	Quality Control
RHEL	Red Hat Enterprise Linux
RO	Radio Occultation (also: GPS-RO, GNSS-RO)
ROM SAF	Radio Occultation Meteorology SAF (formerly GRAS SAF)
GBGP	Ground-Based GNSS Package
SAF	Satellite Application Facility (EUMETSAT)
SAPOS	SAtelliten POSitionierungsdienst der deutschen Landesvermessung (Germany)
SINEX	Solution- (or Software-)INdependent Exchange format
SNR	Signal-to-Noise Ratio
SQL	Structured Query Language
TCP	Transmission Control Protocol
UCAR	University Center for Atmospheric Research (Boulder, CO, USA)
UK	United Kingdom of Great Britain and Northern Ireland
USP	Unique Selling Point
UTC	Universal Time Coordinated
WGS-84	World Geodetic System, 1984
WMO	World Meteorological Organisation
ZHD	Zenith Hydrostatic Delay
ZTD	Zenith Total Delay
ZWD	Zenith Wet Delay

1.4 Definitions

GB-GNSS data products under the E-GVAP project and other data suppliers (such as UCAR) of NRT or offline products:

Data levels:

Level 0:	Raw phase tracking and ancillary data, and other GNSS data before clock correction and reconstruction;
Level 1a:	Reconstructed full resolution excess phase, SNR, amplitude, orbit information
Level 1b:	Zenith total delay, timestamped and annotated with GNSS station location, meta-data and quality information;
Level 2:	Zenith wet delay, integrated water vapour, ancillary meteorological data
Level 3:	Gridded Level 1 and 2 offline products in the form of, e.g., hourly time series, daily or longer means, meta-data, and quality information.

Product types:

NRT product:	data product delivered less than 1.5 hours after measurement;
Offline product:	data product delivered greater than 1 day after measurement;
Reprocessed product:	data product processed consistently over a long dataset.

File format Types:

COST-format:	Text-based format defined by E-GVAP and used for general exchange of GB-GNSS Level 1/2 data. This format is defined in [RD.4];
BUFR:	WMO binary format for dissemination of NRT observational data on the GTS. For GB-GNSS details, see [RD.5];
netCDF:	Unidata binary format for general data storage and exchange. [RD.6]. For GB-GNSS data and documentation on this format, see [RD.7] and [RD.8];
CSV:	A simple text-line-based flat-format. For GB-GNSS data and documentation on this format, see [RD.7].

Note that the ROM SAF does not itself process or provide any GB-GNSS data products; it only maintains the GBGP software to assist in disseminating and using this data type.

2. Overview

This document outlines the first full or operational release of the Ground-Based GNSS Package (GBGP) Version 1.0 or just **GBGP-1**. It describes GBGP structure and the use of the principle library functions, executable tools, supporting scripts and run-time data files provided in the package.

This release is designated 'full' or 'operational' as it has undergone the ROM SAF's formal release procedure which includes code review, a comprehensive test suite, independent (external) beta-testing, a Delivery Readiness Inspection Review and formal approval to release by the ROM SAF Steering Group as fully able to meet operational requirements.

GBGP is a collection of software (provided as source code), supporting build scripts, data files and documentation. Its principle purpose is to encode E-GVAP 'COST-format' files to WMO-standard BUFR format for dissemination to NWP users locally or globally via the GTS. The package also contains several pre-conversion tools to convert other formats, such as netCDF and CSV, to COST-format prior to BUFR encoding. Support for quality control against an SQL database of GNSS station meta-data is also included. Users may wish to integrate a subset of GBGP code into their own software applications, for instance linking the GBGP library to their own code and/or adapting one of the conversion tools to input data a different format.

This document provides a quick guide to building the GBGP package and then assumes that the GBGP software and supporting data files (and external dependency packages) have been successfully built, tested and installed to their target locations as instructed in the GBGP Release Notes [RD.2] and that the appropriate environment variables (notably **`PATH`**, **`MANPATH`**, **`LD_LIBRARY_PATH`**, **`BUFR_LIBRARY`** and **`GBGP_*`**) are correctly defined for the current session, whether for interactive terminal login or via a shell script running as a **`cron`** job for instance.

Illustration 1 shows the GBGP code structure with the command-line toolset and GBGP and third-party library inter-dependencies; Illustration 2 shows the C and Fortran-90 source code modules compiled into the GBGP object library. Modules in square brackets are included or not depending on the Fortran compiler used or the presence (or absence) of certain dependencies. A brief description of the command-line tools and their usage is given in Section 4 and an introduction to the library modules is given in Section 5. The GBGP Reference Manual [RD.3] contains full descriptions and calling details of each and every GBGP Fortran source code main programs and routines.

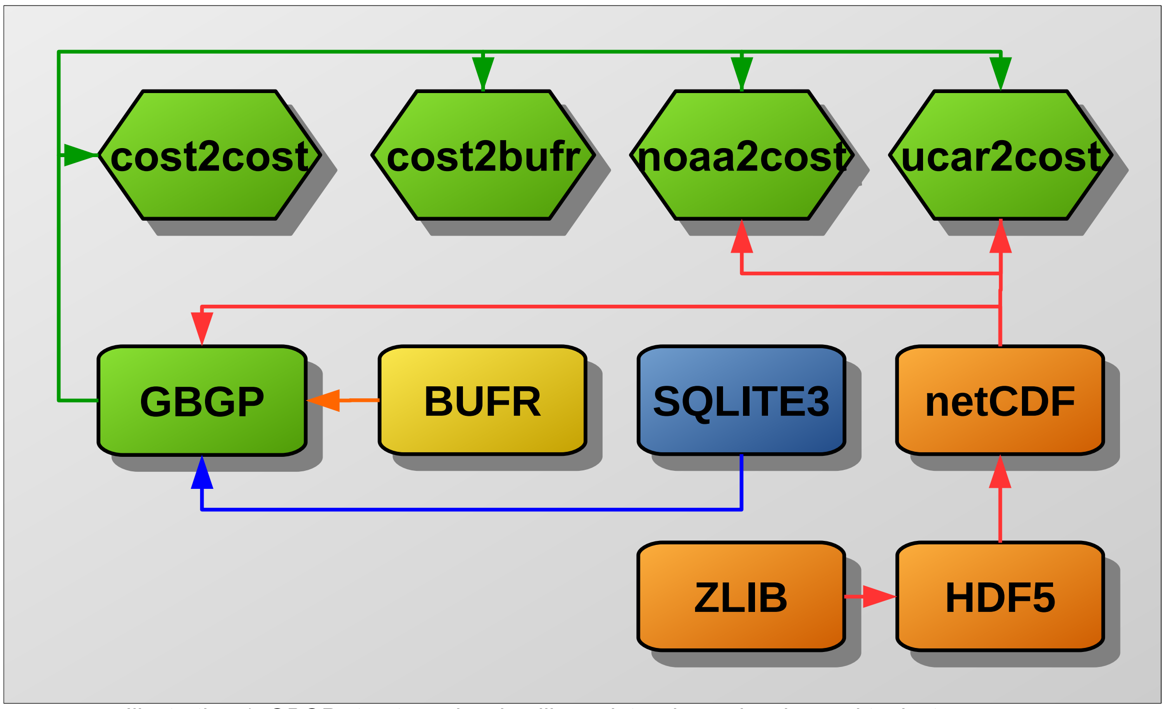


Illustration 1. GBGP structure showing library inter-dependencies and toolset

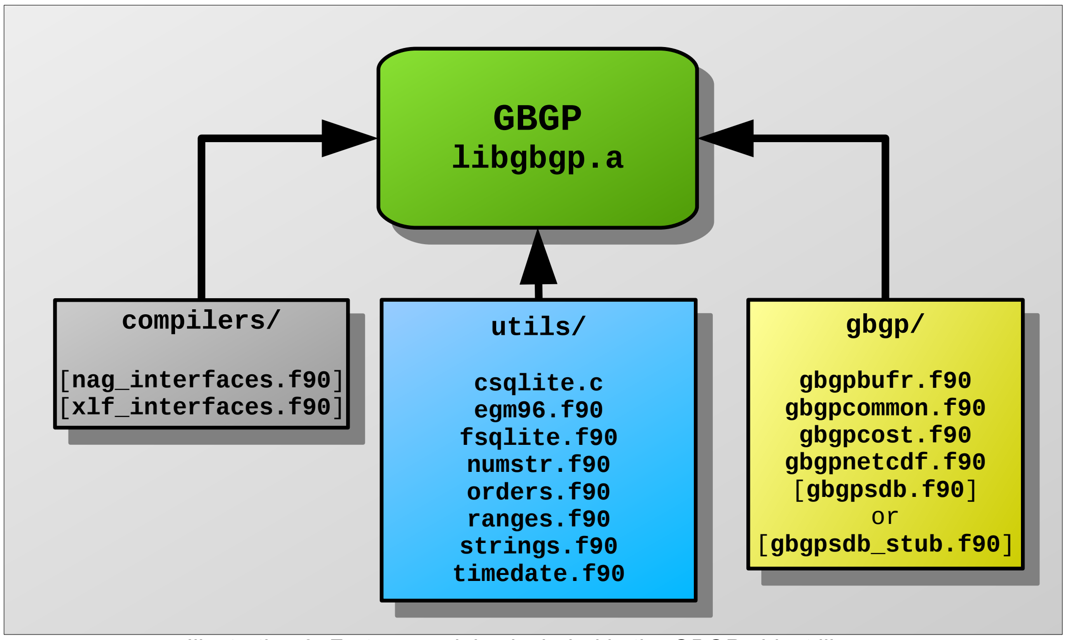


Illustration 2. Fortran modules included in the GBGP object library

3. Quick-start Guide

This section provides a quick start method for building and installing the GBGP software. This does not go into depth and is designed to get a user up and running without needing to know the details. Greater details on the build process can be found in the Release Notes.

Proceed as follows:

1. Check that the compilers are available, GNU **gcc** C compiler and your choice of Fortran compiler. A list of suitable Fortran compilers is found in the Release Notes section 9. To check they are available

   ```
   > gcc -v
   > <compiler> -v
   ```

   This will show what version of the compiler is installed and if not available will have an error. For further details on how to use different compilers see the Release Notes Section 6.4

2. Create a directory for the package, **$HOME/packages/GBGP/** would be a suitable directory and is used in this example.

3. **cd** to this directory and download or copy the main GBGP distribution (**gbgp-1.0.0.tar.gz**), the buildpack distribution (**gbgp_buildpack-1.0.0.tar.gz**), the zlib (e.g. **zlib-1.2.8.tar.gz**), HDF5 (e.g. **hdf5-1.8.18.tar.gz**), netCDF-c (e.g. **netcdf-c-4.4.1.1.tar.gz**), netCDF-fortran (e.g. **netcdf-fortran-4.4.4.tar.gz**), BUFR (e.g. **bufr-24.0.2.tar.gz**) and SQLite (e.g. **sqlite-autoconf-3150200.tar.gz**) tarballs here. Tested versions of these packages are listed in the Release Notes Table 3 and are available on the ROM SAF web site http://www.romsaf.org/gbgp/files.php.

4. unpack the buildpack tarball:

   ```
   > tar -zxvf gbgp_buildpack-1.0.0.tar.gz
   ```

5. A script is provided in the GBGP (buildpack) which can now be run to set up the environment.

   ```
   >  ./setgbgpenv_package.sh
   ```

   This will set up the environment variables required for using GBGP. Among other things, this defines **GBGP_ROOT**, which is the installation target. Users can adjust this to meet their requirements. **$HOME/packages/GBGP/**, the same place the packages were downloaded, would be a suitable choice for **$GBGP_ROOT**, although this is not necessary.

6. If the user has an E-GVAP account then the login details used to access the FTP need to be put into a **.ncftp_FTP** and a **.netrc** file, for the dissemination and database scripts respectively, to run successfully.

   The **.ncftp_FTP** file needs to be created in **$HOME** in this format:

   host <host-URL> (e.g. **ftp.metoffice.gov.uk**)

   user <user-name> (e.g. **gbgnss-<id>**)

   password <password>

   The **.netrc** file also needs to be created in **$HOME** in this format:

   machine <host-URL> login <user-name> password <password>

   These files must be protected by setting R+W access to owner only e.g.:

   ```
   > chmod 600 ~/.ncftp_FTP
   > chmod 600 ~/.netrc
   ```

   Tests 5 and 6 when building GBGP require read/write and read access, respectively, to the E-GVAP FTP hub server. If the necessary login details are not found, these tests will be skipped, but if attempted and access is denied, a **FAIL** will be logged; this is only significant if the user normally has the required access rights.

7. Build and install dependencies and GBGP, in the following order:

   ```
   > ./gbuild_zlib <compiler>
   > ./gbuild_hdf5 <compiler>
   > ./gbuild_netcdf <compiler>
   ```
   (this builds netcdf-c and then netcdf-fortran)
   ```
   > ./gbuild_bufr <compiler>
   > ./gbuild_sqlite <compiler>
   > ./gbuild_gbgp <compiler>
   ```

   which will unpack the GBGP tarball, perform extra checks and save a logfile of the process (e.g. in **`$GBGP_ROOT/<compiler>/gbuild_gbgp.log`**) using all default settings (recommended).

   It is advised that the dependencies are built using the same version of the compiler that will be used for the GBGP software. Following the steps outlines above will meet that requirement. More advanced users may prefer to use external dependency packages that are already installed on their system (by linking them to **`$GBGP_ROOT /<compiler>/lib`**), but for novices it is probably easier and safer to follow the above procedure.

8. Check that the programs have been installed by running:

   ```
   > cd gbgp-1.0.0/tests
   > make check 2>&1 | tee check.log
   ```

   The **`check.log`** file should record all the results of six tests, described in the Release Notes Section 7. It can be compared to **`example_gbgp_test.log`** in the same directory. If all has gone well, **`check.log`** should finish with:

   ```
   Summary of test results:
   TEST 1: cost2cost   PASS
   TEST 2: cost2bufr   PASS
   TEST 3: noaa2cost   PASS
   TEST 4: ucar2cost   PASS
   TEST 5: disseminate PASS
   TEST 6: update database    PASS
   ```

   although either or both of the last two tests may not have been run if the user does not have an E-GVAP account (see point 6 above).

9. In case of difficulties, the user should examine the build log files and the make check log file for clues. Further advice can be found in the Release Notes Section 6.

4. Tool set

The Fortran-90 source code files for these tools can be found in the GBGP sub-directory **`tools/`**

4.1 `cost2bufr`

The **`cost2bufr`** command-line tool is the principle component of GBGP; as the name implies, it encodes the contents of a COST-format [RD.4] file to WMO FM94 (BUFR) using the published template for this data type [RD.5]. By default, the tool inserts a 10-byte IP leader and GTS abbreviated routing headers (ARH) and trailers so that resulting one or more complete *bulletins* in the BUFR output file can be transmitted to other NWP users via the WMO GTS network via standard TCP/IP protocols. Default settings are appropriate for use with the Met Office GTS node; options can be used to fine-tune the IP leader and ARH for other nodes.

A summary of usage can be printed using the help option:

```
> cost2bufr -h
```

Detailed usage and examples can be found in the 'man' page:

```
> man cost2bufr
```

The full command line is:

```
> cost2bufr cost_file [cost_file...] [-b csn_file] [-c code]
            [-ga|-gi|-gh] [-o bufr_file] [-s status] [-t time]
            [-m] [-d] [-h|?] [-v]
```

Input:

One or more files which must be in COST-format V2.x [RD.4] (plain text, not compressed). While reading V1.0 files is still supported, these will be so old as to be unlikely to be useful for applications using BUFR. Shell-expanded wildcards are also supported.

Options:

`-b`	file name for a saved (cached) bulletin (channel) sequence number between runs
`-c`	originating centre code value. Encoded into BUFR Section 1 and used to look-up the ICAO code for the originator field of the ARH.
`-d`	debug mode: outputs additional diagnostics to **`stdout`**
`-ga`	generate routing headers with alternate IP19 leaders
`-gh`	don't generate GTS abbreviated routing headers (ARH)
`-gi`	generate ARH without IP leaders
`-h`	write summary usage help
`-m`	skip meta-data checks against database; only do basic Q/C validity checks
`-o`	BUFR output file name
`-s`	force file status when generating routing headers
`-t`	don't encode data older than 'time' ago (hh:mm)
`-v`	write program version ID to **`stdout`**

Defaults:

Input file name:	**`cost.dat`**
Output file name:	**`gbgp.bfr`**
Originating centre code:	**`074`** (ICAO code **`EGRR`** – UK Met Office, Exeter (RSMC))
GTS routing headers:	file-dependent status ARH with IP10 leaders
Bulletin sequence number:	starts at **`001`**
Reject time difference:	**`23:50`** unless **`-gh`** option, when **`0:00`** (no filtering on age)
Meta-checks:	check for duplicated station IDs

Output:

One output BUFR message per hour of input data, including multiple stations if present, but with no more than 500 observations (timestamped ZTD samples) per message.

Environment:

`BUFR_LIBRARY`	path to run-time BUFR tables (see below)
`GBGPMSG_MODE`	default message mode - one of: **`INFO`** (normal mode), **`DEBUG`**, **`WARNINGS`**, **`ERRORS`** or **`QUIET`**. May be abbreviated to the initial letter; case insensitive
`GBGP_DATA`	path to supporting GBGP run-time data files Optional: defaults to **`~/data/gbgp`**
`GBGP_SDB`	path to meta-data SQL database. Optional: defaults to **`GBGP_DATA/gbgnss.sdb`** Set to **`NONE`** to skip database Q/C checks
`GBGP_ISOFIX`	path to file used for ISO-country Q/C (mapping broken country names given in some COST-format files to their correct ISO-name) Optional: defaults to **`GBGP_DATA/isofix.dat`**
`GBGP_SIDFIX`	path to file used for GNSS ID Q/C (mapping known duplicate IDs so some arbitrary but unique ID) Optional: defaults to **`GBGP_DATA/sidfix.dat`**

Other run-time dependencies:

BUFR look-up tables in **`BUFR_LIBRARY`** (installed with the Met Office BUFR dependency package):

`bufr_tableb`	parameter descriptor details (bit-width, scaling factor, etc)
`bufr_tabled`	master and sub-sequence descriptor list of parameter descriptors
`bufr_orgcentre`	table of originating centres and sub-centres required to generate the ARH and encoded into BUFR Section 1 (see [RD.5])

Other tables in this path or not required for the encoding process.

Abbreviated routing headers (ARH):

The full generated ARH depends on:

- the bulletin (channel) sequence number;
- the geographical area covered by the encoded observations;
- file status (operational, demonstration or test);
- originating centre ICAO code (default **`EGRR`**) and
- the day-of-month & time representative of the encoded observations.

The full ARH for this data type is given in [RD.5] (see also the **`cost2bufr`** 'man' page), but the following summarizes the mapping between file status and the significant part of the ARH (where **`i`** is the area designator letter code, potentially any one one of **`[A-L,N,S,T,X]`**):

OPER → **`ISXi14`**	general GTS routing to any NWP centre requesting these headers
DEMO → **`ISXi15`**	limited GTS routing to approved NWP centres for evaluation only
TEST → **`ISXi16`**	not for GTS dissemination

All bulletins are routed to the internal Met Office meteorological observations database, the 'MetDB' which is the source for all E-GVAP monitoring, with a sub-set assimilated into our NWP system.

4.2 cost2cost

The **`cost2cost`** command-line tool is a converter, format-checker and quality-control/filter utility depending on it's use. Essentially, it reads any version (V1.0, V2.x) COST-format file and writes the data to the latest supported version (currently V2.2).

Any parameters in the latest version not defined in an older version are assigned 'unknown' or 'missing data' values. In this mode, it can be used to convert older formats to the latest version, with basic (e.g. range-checking) quality control; if the SQLite database is enabled, then additional Q/C such as duplicate GNSS station ID detection can be performed and missing data can be replaced with valid data where known from other sources. Any station data failing Q/C is rejected and not written to the output file. If the input file is already the latest version, **`cost2cost`** can be used to filter out any 'bad' data and as long as the input was correctly formatted, all data passing Q/C will be unchanged in the output.

On another level, **`cost2cost`** can be used to quickly check that a file (typically generated by a new AC and submitted for pre-checking before moving to live uploads) is correctly formatted according the file specification [RD.4]. While this isn't foolproof, it does generally detect mis-formatted data such as incorrect field widths (usually triggering range-check errors as column-oriented values will often be read out of sync), missing or extra lines (blank or missing lines are a common problem with first attempts). Having removed formatting problems, a sample file can then be checked for accidental duplicate GNSS station IDs, or for existing stations processed by other ACs that their station meta-data is consistent (or not).

Finally, **`cost2cost`** can be used to auto-generate a file name based on the file's contents (with or without actually creating the output file); this can be used to merely check that the original file name is consistent with the content (especially for checking the latest file name scheme), or to purposely rename a file to make it so – either by reading and re-writing a new file, or capturing the new name and explicitly renaming it.

A summary of usage can be printed using the help option:

```
> cost2cost -h
```

Detailed usage and examples can be found in the 'man' page:

```
> man cost2cost
```

The full command line is:

```
> cost2cost infile [infile...] [-d] [-h|?] [-m] [-n] [-p path]
            [-q] [-s status] [-v]
```

Input:

One or more files which must be in any supported version (V1.0, V2.x) of the COST-format (see [RD.4] for the latest version). Shell-expanded wildcards are also supported.

Options:

`-d`	debug mode: outputs additional diagnostics to **`stdout`** (overrides **`-q`**)
`-h`	write summary usage help
`-m`	skip meta-data checks against database; only do basic Q/C validity checks
`-n`	name-only mode; print the generated file name but do not output a COST-format file to disk (implies **`-q`**)
`-p`	path (directory) for output files(s); the file names themselves are auto-generated
`-q`	quiet mode: suppress all messages (except the new file name with **`-n`**) to **`stdout`**
`-s`	force a file status indicator – one of **`OPER`**, **`DEMO`** or **`TEST`** May be abbreviated to the initial letter; case insensitive.
`-v`	write program version ID to **`stdout`**

Defaults:

Input file name:	**`cost.dat`**
Output path:	current working directory
Status:	**`UNKNOWN`**
Meta-checks:	check for duplicated station IDs
Name mode:	output COST-format file(s)
Quiet mode:	normal output

Output:

One output COST-format file [RD.4] per input file. Output file names are automatically generated based on the input file contents, possibly modified by command-line options.

Environment:

`GBGPMSG_MODE`	default message mode - one of: **`INFO`** (normal mode), **`DEBUG`**, **`WARNINGS`**, **`ERRORS`** or **`QUIET`**. May be abbreviated to the initial letter; case insensitive
`GBGP_DATA`	path to supporting GBGP run-time data files Optional: defaults to **`~/data/gbgp`**
`GBGP_SDB`	path to meta-data SQL database. Optional: defaults to **`GBGP_DATA/gbgnss.sdb`** Set to **`NONE`** to skip database Q/C checks
`GBGP_ISOFIX`	path to file used for ISO-country Q/C (mapping broken country names given in some COST-format files to their correct ISO-name) Optional: defaults to **`GBGP_DATA/isofix.dat`**
`GBGP_SIDFIX`	path to file used for GNSS ID Q/C (mapping known duplicate IDs so some arbitrary but unique ID) Optional: defaults to **`GBGP_DATA/sidfix.dat`**
`EGM96_GRID`	path to the EGM96 gridded undulation s file. (default: **`~/data/emg96/egm96_grid.dat`**) Input files may contain only the ellipsoid (WGS-84) station height but not a valid (usually set to the 'missing data' flag value) geoid (EGM96) height (taken to be height AMSL). This file is used to calculate the equivalent geoid height using the given GNSS station WGS-84 lat/lon/ht coordinates (if all are themselves valid).

4.3 noaa2cost

The **`noaa2cost`** command-line tool is a pre-converter utility and as the name implies, it converts the contents of a NOAA netCDF or CSV file [RD.6][RD.7] to a COST-format [RD.4] file. The output file can then be encoded to BUFR using the **`cost2bufr`** tool and/or collected with COST-format files from other sources. Note that since there is limited meta-data contained in the NOAA CSV format, several parameters in the output COST-format file will be missing or set to (assumed) default values. If enabled, then meta-data from other ACs can in principle be substituted. However, since most of these are not included in the BUFR template, this is not a serious limitation if BUFR is the desired final target format.

A summary of usage can be printed using the help option:

```
> noaa2cost -h
```

Detailed usage and examples can be found in the 'man' page:

```
> man noaa2cost
```

The full command line is:

```
> noaa2cost infile [infile...] [-d] [-h|?] [-m] [-n] [-p path]
            [-q] [-s status] [-v]
```

Input:

One or more files which must be in NOAA netCDF or CSV format [RD.6][RD.7] and having the file type **`CDF`** or **`CSV`** as part of the file name. Both formats can be specified in a single conversion. Shell-expanded wildcards are also supported.

Options:

`-d`	debug mode: outputs additional diagnostics to **`stdout`** (overrides **`-q`**)
`-h`	write summary usage help
`-m`	skip meta-data checks against database; only do basic Q/C validity checks
`-n`	name-only mode; print the generated file name but do not output a COST-format file to disk (implies **`-q`**)
`-p`	path (directory) for output files(s); the file names themselves are auto-generated
`-q`	quiet mode: suppress all messages (except the new file name with **`-n`**) to **`stdout`**
`-s`	force a file status indicator – one of **`OPER`**, **`DEMO`** or **`TEST`** May be abbreviated to the initial letter; case insensitive.
`-v`	write program version ID to **`stdout`**

Defaults:

Input file name:	none – at least one file name required
Output path:	current working directory
Status:	**`TEST`**
Meta-checks:	check for duplicated station IDs
Name mode:	output COST-format file(s)
Quiet mode:	normal output

Output:

One output COST-format file [RD.4] per input file. Output file names are automatically generated based on the input file contents, possibly modified by command-line options.

Environment:

`GBGPMSG_MODE`	default message mode - one of: **`INFO`** (normal mode), **`DEBUG`**, **`WARNINGS`**, **`ERRORS`** or **`QUIET`**. May be abbreviated to the initial letter; case insensitive.
`GBGP_DATA`	path to supporting GBGP run-time data files Optional: defaults to **`~/data/gbgp`**
`GBGP_SDB`	path to meta-data SQL database. Optional: defaults to **`GBGP_DATA/gbgnss.sdb`** Set to **`NONE`** to skip database Q/C checks
`EGM96_GRID`	path to the EGM96 gridded undulation s file. (default: **`~/data/emg96/egm96_grid.dat`**) NOAA netCDF & CSV files contain only station height AMSL; this file is used to calculate the equivalent ellipsoidal (WGS-84) height.

Other run-time dependencies:

`GPSMETStation.txt`	Optional, but expected in in path **`GBGP_DATA`**. This file contains a list of the current 'known GPSMET stations'. A new station ID in a NOAA CSV file (i.e. not yet in the SQL database) is searched for in this list so that certain missing meta-data (such as station name and country information) can be included in the output COST-format file. If this file is not found, or the new station ID is not included in it, **`noaa2cost`** will just set assumed default (or 'unknown') values. Not needed for NOAA netCDF files, which contain the required meta-data.

Notes:

1. NOAA GB-GNSS data converted to BUFR should not be disseminated on the GTS except by NOAA themselves[4].

4.4 ucar2cost

The **`ucar2cost`** command-line tool is a pre-converter utility and as the name implies, it converts the contents of a UCAR netCDF file [RD.6][RD.8] to a COST-format [RD.4] file. The output file can then be encoded to BUFR using the **`cost2bufr`** tool and/or collected with COST-format files from other sources. Note that since there is limited meta-data contained in the UCAR netCDF, several parameters in the output COST-format file will be missing or set to (assumed) default values (or – like as **`noaa2cost`** – potentially filled in from meta-data provided by other ACs). However, since most of these are not included in the BUFR template, this is not a serious limitation if BUFR is the desired final target format.

A summary of usage can be printed using the help option:

```
> ucar2cost -h
```

Detailed usage and examples can be found in the 'man' page:

```
> man ucar2cost
```

The full command line is:

```
> ucar2cost infile [infile...] [-d] [-h|?] [-m] [-n] [-p path]
            [-q] [-s status] [-v]
```

Input:

One or more files which must be in UCAR netCDF format [RD.6][RD.8]. Shell-expanded wildcards are also supported.

Options:

`-d`	debug mode: outputs additional diagnostics to **`stdout`** (overrides **`-q`**)
`-h`	write summary usage help
`-m`	skip meta-data checks against database; only do basic Q/C validity checks
`-n`	name-only mode; print the generated file name but do not output a COST-format file to disk (implies **`-q`**)
`-p`	path (directory) for output files(s); the file names themselves are auto-generated
`-q`	quiet mode: suppress all messages (except the new file name with **`-n`**) to **`stdout`**

[4]*NOAA/ESRL have contracted a private company to process their GB-GNSS data for NOAA's own internal use only. Notice was given that GTS dissemination would formally cease on 1 September 2016, although there was a period of grace after that date; GTS data actually stopped on 3 October 2016 and NetCDF/CSV stopped appearing on their FTP server on 29th November 2016. Non-NOAA users are expected to pay a fee for access. The **`noaa2cost`** tool is provided for processing pre-downloaded historic data and support might be withdrawn from future releases of GBGP unless NOAA reverse their data access policy.*

`-s`	force a file status indicator - one of **`OPER`**, **`DEMO`** or **`TEST`** May be abbreviated to the initial letter; case insensitive.
`-v`	write program version ID to **`stdout`**

Defaults:

Input file name:	none – at least one file name required
Output path:	current working directory
Status:	**`TEST`**
Meta-checks:	check for duplicated station IDs
Name mode:	output COST-format file(s)
Quiet mode:	normal output

Output:

One output COST-format file [RD.4] per input file. Output file names are automatically generated based on the input file contents, possibly modified by command-line options.

Environment:

`GBGPMSG_MODE`	default message mode - one of: **`INFO`** (normal mode), **`DEBUG`**, **`WARNINGS`**, **`ERRORS`** or **`QUIET`**. May be abbreviated tot he initial letter; case insensitive.
`GBGP_DATA`	path to supporting GBGP run-time data files Optional: defaults to **`~/data/gbgp`**
`GBGP_SDB`	path to meta-data SQL database. Optional: defaults to **`GBGP_DATA/gbgnss.sdb`** Set to **`NONE`** to skip database Q/C checks
`EGM96_GRID`	path to the EGM96 gridded undulation s file. (default: **`~/data/emg96/egm96_grid.dat`**) UCAR netCDF files contain only station height AMSL; this file is used to calculate the equivalent ellipsoidal (WGS-84) height.

Other run-time dependencies:

`conus.table` Optional, but expected in in path **`GBGP_DATA`**. This file contains a table with the current 'known CONUS stations' listing. A new station ID in a UCAR netCDF file (i.e. not yet in the SQL database) is searched for in this table so that certain missing meta-data (such as station name and country information) can be included in the output COST-format file. If this file is not found, or the new station ID is not included in it, **`ucar2cost`** will just set assumed default (or 'unknown') values.

Notes:

1. UCAR data is freely available to download but is limited to local evaluation; data should not be passed to third parties in native netCDF or any converted format unless expressly agreed with UCAR. This includes COST-format files not being posted to the E-GVAP hub server and certainly not to put BUFR on the GTS. UCAR (COSMIC Office) may arrange to encode their SuomiNet/CONUS network data and disseminate the BUFR via the GTS themselves in the future.

4.5 decbufr

Although the **`decbufr`** command-line tool is component of the MetDB BUFR package and not part of the GBGP distribution, we mention it here as many users have said that they found it useful for checking any arbitrary BUFR message (as long as it was correctly encoded and all used and valid descriptors are present in the installed MetDB BUFR tables), including of course GB-GNSS data encoded with **`cost2bufr`**.

While there are many such generic decoders (and there is a simple one included in the base MetDB kernel library package), the USP of **`decbufr`** is that it maps code values to interpreted, human-readable text equivalents for all BUFR header Section 1 elements only or as well as for data in Section 4 and can decode a sub-set of multiple BUFR messages within a single file.

Summary help is available with the usual **`-h`** option and for fuller details, see the **`README`** file in **`bufr-<ver>/extra`** or the 'man' page. **`Decbufr`** uses look-up tables files in the path defined by EV **`BUFR_LIBRARY`**.

5. The GBGP library

The GBGP library (i.e. non-tools) source code provided in the package is split into a number of files (many being Fortran-90 modules containing related routines and parameter definitions). There is but a single C source code file which provides an F90 interface to the C-based SQLite dependency library. The source code files are further split into three directories (see Illustration 2) according to purpose. Together, they provide the 'heavy lifting', simplifying the code in the individual command-line tools to that necessary only for that job.

5.1 Compiler-dependent

GBGP sub-directory: **`compilers/`**

These files provide some wrappers to non-ISO standard – but nevertheless industry-common – system routines such as **`EXIT()`** for certain Fortran compilers which keeps the main application code compiler-independent. In general, these wrappers trap 'standard' calls and convert them to their vendor-specific equivalents. They are only compiled and linked with the tools when using the associated vendor's Fortran compiler and not otherwise (strictly, the wrapper object file is not actually included in the GBGP object library but passed directly to the linker; this avoids the vendor's library version being resolved by the linker before the wrapper version in GBGP library). This code is PES (see next section).

`nag_interfaces.f90`	Used only when building with the NAGWare Fortran-90 (**`nagfor`**) compiler on Linux systems
`xlf_interfaces.f90`	Used only when building with the IBM Fortran-90 compiler (**`xlf95`**) compiler on IBM AIX (HPC/Supercomputer) systems

5.2 Utilities

GBGP sub-directory: **`utils/`**

The files here contain general utility code, not specific to GBGP. In EUMETSAT ROM SAF-speak this code has been approved by the Steering Group as having the status of 'Pre-Existing Software' (PES). ***This means that any PES item is not owned by EUMETSAT but remains the property (and copyright) of the originator, and has a licence implied or expressed for open use, or at least no more restrictive that the terms of the GBGP User Licence. Users must abide by any licence terms associated with any PES item.***

`csqlite.c`	provides a C interface between **`fsqlite.f90`** and the C-based SQLite dependency library. Not compiled if SQLite is not installed
`fsqlite.f90`	provides a Fortran-90 API to the C-based SQLite dependency library via the **`csqlite.c`** interface. Not compiled if SQLite is not installed
`egm96.f90`	a module used to convert GNSS ellipsoidal (WGS-84) station heights to geoidal (EGM96) heights by interpolating a gridded 'undulations' 2-D field at the latitude & longitude of the GNSS station. For practical NWP purposes the EGM96 geoidal height can be taken to represent the station height above mean sea level. The grid file is found via the environment variable **`EGM96_GRID`** (default: **`~/data/egm96/egm96_grid.dat`**)
`numstr.f90`	a module for converting numerical values to left-justified strings to simplify message printing.

`orders.f90`	a module for ordering (via a sorted index with duplicate value flagging) 1-D arrays of strings, integers, floats or doubles.
`ranges.f90`	a module to range-check a value of any numeric type with optional value substitution/flagging if detected as out-of-range.
`strings.f90`	a module providing several string-handling routines.
`timedate.f90`	a module providing date/time handling routines, including various representations (string, array, Julian Day, UNIX, GPS, ...), base conversions, string parsing and formatting from/to several standard forms.

5.3 GBGP-specific

GBGP sub-directory: **`gbgp/`**

These files contain the code written specifically for GBGP. As such it is owned by EUMETSAT and is provided under the terms of the GBGP User Licence which users will have needed to have signed before downloading GBGP.

Apart from the common module, each source file contains the code to interface with one specific file type, which makes the structure flexible and simplifies the application tools as they only need to reference the module for the file type(s) it needs to process. (We follow the guidance that a piece of code should do only one job, and do that job well, not try to be a jack-of-all-trades.)

`gbgpcommon.f90`	GBGP module containing some routines common across the package, such as a message-printing routine, and fixed parameter value definitions
`gbgpbufr.f90`	GBGP module providing BUFR-related routines [RD.5]
`gbgpcost.f90`	GBGP module providing an I/O interface for COST-format file [RD.4]
`gbgpnetcdf.f90`	GBGP module providing netCDF-related routines [RD.6] The module currently only contains an error message routine, but has potential for more should a standard netCDF definition be developed to replace the present COST-format for data exchange. [Today that seems unlikely as a development of the SINEX-TROPO format is being proposed instead]
`gbgpsdb.f90`	GBGP module providing a simplified interface to the SQLite library (via **`fsqlite.f90`** and **`csqlite.c`**) specifically for the GB-GNSS meta-data database. This module is only compiled and included in the GBGP object library if the SQLite library is installed. The database file is found via the environment variable **`GBGP_SDB`** (default: **`GBGP_DATA/gbgnss.sdb`** with **`GBGP_DATA`** defaulting to **`~/data/gbgp`**)
`gbgpsdb_stub.f90`	GBGP module providing a dummy interface to the SQLite meta-data database; compiled instead of **`gbgpsdb.f90`** if SQLite is not installed. This stub provides the same API but does not call any lower-level routines (from **`fsqlite.f90`**) and behaves as if **`GBGP_SDB=NONE`** had been set ('database not available' return status).

6. Shell scripts

Two shell (Bash) scripts are provided; one which is manages file dissemination to various targets (designed for internal Met Office use, but may be useful for others if suitably modified) and one to keep a local copy of the master GB-GNSS database up to date.

GBGP sub-directory: **`scripts/`**

6.1 `gbgp_disseminate.sh`

This script takes GB-GNSS data as a COST-format file (plain text or gzipped) and disseminates the data to various targets (using FTP) as either a compressed COST-format file or as a BUFR-encoded file of bulletins with appropriate abbreviated routing headers according the input file status (see **`cost2bufr`** in Section 4.1)

In the following the **HUB** server target is either the *FTPPUBLIC* Internet-facing server for data exchange within the E-GVAP project, or *Dart*, an internal routing service which may indirectly upload files to FTPPUBLIC (and elsewhere). The **GTS** target is normally the Met Office *MetSwitch* GTS node; this will route bulletins to the external GTS and/or the internal operational observations database 'MetDB' according to the bulletins' ARH, but Dart can also be an intermediate target (a legacy option).

A summary of usage can be printed using the help option:

```
> gbgp_disseminate.sh -h
```

The full command line is:

```
> gbgp_disseminate.sh  [<cost-file>] [GTS] [HUB]
                       [-h] [-s OPER|DEMO|TEST] [-v]
```

Input:

A COST-format V2.x file [RD.4] (plain text or **`gzip`**-compressed).

Keywords:

`GTS`	If present, the GTS target is enabled (even if **`GBGP_GTS`** is valid)
`HUB`	If present, the HUB target is enabled (even if **`GBGP_HUB`** is valid)

Options:

`-d`	sets debug mode – the script is run normally except that the actual FTP upload command line is echoed to **`stdout`** instead of being executed and the **`cost2bufr -d`** debug option is set
`-h`	write summary usage help
`-s`	force file status; one of **`OPER`**, **`DEMO`** or **`TEST`**
`-v`	write script version ID to `stdout`

Defaults:

Input file name:	**`cost.dat`**
GTS target:	disabled
Hub target:	disabled

Output:

If enabled, the (compressed) COST-format file is uploaded to the hub server
If enabled, the (encoded) BUFR file is uploaded to the GTS server, which in turn will route the bulletins to the GTS and/or the MetDB depending on the ARHs generated by **`cost2bufr`**.

Environment:

The following are *in addition* to the environment variables shown for **`cost2bufr`**

`GBGP_GTS`	target GTS/MetDB server ID, for example **`METSW`** (MetSwitch GTS node). As provided, this target is only valid for internal Met Office use. If **`NONE`** (or the **`GTS`** keyword is not present), then uploads to this target are disabled.
`GBGP_HUB`	target hub server ID, for example **`FTP`** or **`DART`** (**`FTPPUBLIC`** or Dart servers) As provided, this target is only valid for internal Met Office use. If **`NONE`** (or the **`HUB`** keyword is not present), then uploads to this target are disabled.
`GBGP_EMAIL`	a valid email address; any FTP failure messages will be sent to this user. If set to **`NONE`** (default), no failure messages will be sent.
`GBGP_OCEN`	defines the Originating/Generating (encoding) Centre BUFR code. Default: **`74`** (EGRR/Met Office, Exeter)
`GBGP_MSGLOG`	path to a message log file. A message is a single line formatted Tivoli-style, recording for instance a GTS/MetDB FTP copy failure. If undefined or **`NONE`**, no messages are written. Default: **`NONE`**

Other run-time dependencies:

BUFR look-up tables and related files as described for **`cost2bufr`**.
The script uses the **`ncftpput`** tool from the NcFTP family. This does not use the common **`.netrc`** file to specify login credentials, but an arbitrary file specified on the command line. We use the file name template **`~/.ncftp_<ID>`** where **`<ID>`** is the server ID defined by one of the above environment variables, for instance **`~/.ncftp_FTP`**. This file is formatted as:

```
host <host-URL>          (e.g. ftp.metoffice.gov.uk)
user <user-name>         (e.g. gbgnss-<id>)
password <password>
```

These files must be protected by setting R+W access to the owner only, e.g.:

```
> chmod 600 ~/.ncftp_*
```

6.2 gbgp_updsdb.sh

If the SQLite GB-GNSS database is to be used for quality-control purposes (principally duplicate station ID detection) as part of the pre-conversion and/or BUFR encoding, then to be useful it must be kept up to date and in sync with the live master version which is updated with every incoming COST-format file. A snapshot of the master database (as both a binary SQLite file and as a compressed SQL text file dump is uploaded to the E-GVAP hub server (FTPPUBLIC) several times per day (currently every 6 hours from 03:00 UTC)[5]. This script downloads the latest snapshot dump file from the server to replace a local working version.

A summary of usage can be printed using the help option:

```
> gbgp_updsdb.sh -h
```

The full command line is:

```
> gbgp_updsdb.sh [-f] [-h] [-v]
```

[5] *This requires login access; consideration could be given to also hosting this dump file openly on the ROM SAF website.*

Input:

None

Options:

`-f`	force database rebuild from an existing dump if not downloaded
`-h`	write summary usage help
`-v`	write script version ID to **`stdout`**

Defaults:

Database:	not updated if download fails

Output:

Updated local copy of the SQLite database

Environment:

`GBGP_DATA`	path to supporting GBGP run-time data files Defaults to **`~/data/gbgp`**
`GBGP_SDB`	path to meta-data SQL database Defaults to **`GBGP_DATA/gbgnss.sdb`**

Other run-time dependencies:

The script uses **`wget`** as the FTP download client, the **`gunzip`** utility to decompress the downloaded dump file and the **`sqlite3`** command-line tool to rebuild the binary database.

The script requires a .netrc file with appropriate E-GVAP user login details to be able to access the location of the the database. The .netrc file file is formatted as:

machine <host-URL> login <username> password <password>

(e.g. **machine `ftp.metoffice.gov.uk` login gbgnss-<id> password <password>)**

7. COST-format file I/O API

The **`gbgpcost.f90`** module (see Section 5.3) provides a convenient API to read, write and quality-control COST-format files from a Fortran-90 application. The interface mimics the *vfile* structure of the COST-format, where a *vfile* contains all of the timestamped ZTD and related observations for the nominal period for one GNSS station (*vfile-data*), plus meta-data on that station and the processing (*vfile-header*). For normal near-real time hourly or sub-hourly files, each COST-format file contains an arbitrary number of *vfiles*, one per GNSS station in the network analysis all processed by one analysis centre (AC) for the specified epoch. The COST-format specification document [RD.4] describes this structure, the parameters and their technical formatting in full detail.

This section just gives a brief overview of the principle API routines; for details, see the GBGP Reference Manual [RD.3] or the code itself in source file **`gbgpcost.f90`**. The code for **`cost2cost.f90`** (and the other GBGP tools) also serves to illustrate practical usage of the API, and the tools may act as templates for other user-applications which need to read or write COST-format files.

7.1 Open a COST-file

A COST-format file on disk must first be opened for reading or writing. If writing, and the file already exists, it will be overwritten, else a new file will be created. There is currently no option to append to an existing file.

Prototype:
```
      USE GBGPCOST
      CHARACTER (LEN=n) :: COSTname, mode
      INTEGER :: COSTunit, status
      CALL GBGP_OpenCOST ( COSTname, mode, COSTunit, status )
```

Inputs:

`COSTname`	COST file path+name (default if blank: **`cost.dat`**)
`mode`	file open mode; one of: - **`'READ'`** read only (default) - **`'WRITE'`** write (replace any existing, or create a new file) Case insensitive & may be truncated to initial letter

Outputs:

`COSTname`	final COST file path+name (default may be substituted)
`COSTunit`	assigned COST file I/O stream unit (10-999). **`0`** if the file could not be opened.
`status`	return status; one of: **`0`** : OK **`1`** : File not found (READ mode) **`2`** : Open error (e.g. no permissions)

7.2 Close a COST-file

When a COST-format disk file is finished with, closing it is optional, but it is good practice to explicitly close it. This ensures that any buffers are flushed (when writing) and the stream I/O unit is released.

Prototype:
```
      USE GBGPCOST
      INTEGER :: COSTunit, status
      CALL GBGP_CloseCOST ( COSTunit, status )
```

Inputs:

`COSTunit`	COST file unit (as returned by **`GBGP_OpenCOST()`**)

Outputs:

`status`	return status; one of: **`0`** : OK (file was successfully closed or was not open) **`2`** : Close error

7.3 Read a COST-format file

While routines exist to read or write an entire COST-format disk file to/from memory (one *vfile* at a time, using the routines described here), because there may potentially be a very large number, and in general *vfiles* are independent, it is preferable for an application – having opened the file – to process *vfile*-by-*vfile* in a loop, then close the file. **`GBGP_ReadVFile()`** is designed to be called in a loop, reading and returning the next *vfile* on each call until there are no more in the input disk file. The routine currently supports COST-format versions 1.0, 2.0, 2.1 and 2.2. While data not technically formatted to the strict requirements of [RD.4] may trigger an I/O (disk file or internal read) or unsupported version error, this routine does not perform any explicit quality control.

Prototype:

```
USE GBGPCOST
TYPE (COSTvfile) :: vfile
INTEGER :: COSTunit, status
CALL GBGP_ReadVFile ( COSTunit, vfile, status )
```

Inputs:

`COSTunit`	COST file unit (as returned by **`GBGP_OpenCOST()`**)

Outputs:

`vfile`	COST-format *vfile* (derived type – see Section 7.6)
`status`	return status; one of: **`-1`** : End-of-file (no more *vfiles* – returned *vfile* is invalid) **`0`** : OK **`2`** : I/O error – *vfile* incomplete **`3`** : Unsupported format version

7.4 Write a COST-format file

As for reading, if possible, process and write out COST-format data *vfile*-by-*vfile* in a loop having opened the file for writing, and close when finished. **GBGP_WriteVFile()** is designed to be called in a loop, writing the current *vfile* to disk on each call. The routine currently supports COST-format version 2.2 only. This routine does not perform any explicit quality control, but it does check the overall Q/C and sample Q/C summary values; if the former is non-zero, the *vfile* will not be written at all, and if the sample Q/C is non-zero then the sample is not written. If the user wishes to still output such bad data (not recommended!), then the relevant Q/C flags & summary values need be set to zero, or do not call the Q/C routine.

Prototype:

```
USE GBGPCOST
TYPE (COSTvfile) :: vfile
INTEGER :: COSTunit, status
CALL GBGP_WriteVFile ( COSTunit, vfile, status )
```

Inputs:

`COSTunit`	COST file unit (as returned by **`GBGP_OpenCOST()`**)

`vfile`	COST-format *vfile* (derived type – see Section 7.6)

Outputs:

`status`	return status; one of: **`0`** : OK **`2`** : I/O error – *vfile* not written **`3`** : Q/C error – *vfile* not written

7.5 Quality-control a *vfile*

The quality control (Q/C) tests applied by this subroutine are described in Section 8 and the rôle of the GNSS meta-data database in this procedure is described in Section 9. Performing this Q/C is optional but highly recommended when processing arbitrary COST-format files from a variety of sources. It is left to the application to take appropriate action if a Q/C problem is flagged.

Prototype:

```
USE GBGPCOST
TYPE (COSTvfile) :: vfile
INTEGER :: qstatus
CALL GBGP_QCVFile ( vfile, qstatus, check_name, replace_name )
```

Inputs:

`vfile`	COST-format *vfile* (derived type – see Section 7.6)
`check_name`	(optional) if **`.TRUE.`** (default) check the *vfile* station name against that in the database and report if significantly different. If **`.FALSE.`** silently skip this check. Use this option for instance when the station name is not relevant, such as BUFR encoding.
`replace_name`	(optional) if **`.TRUE.`** (default) replace *vfile* station full name with the database version if they differ. If **`.FALSE.`**, keep the *vfile* name.

Outputs:

`vfile`	potentially updated COST-format *vfile*
`qstatus`	quality-control return status; one of: **`0`** : OK (no critical issues) **`1`** : one or more header elements failed Q/C **`2`** : headers OK, but one or more sample elements failed Q/C **`VFile%QC`** flag bits show which (if any) tests failed.

Q/C flag bits:

None, one or several flag bits may be set to indicate failed test results (unset implies a pass for that test):

`0`	Invalid station ID
`1`	Invalid first sample date/time
`2`	Invalid station latitude
`3`	Invalid station longitude
`4`	Invalid station height
`5`	Invalid processing centre ID
`6`	Duplicate station ID
`7`	No valid samples

7.6 The COST-format derived types

A complete COST-format *vfile* (see specification document [RD.4]) consists of a header (containing meta-data) and data (the time-stamped ZTD and related observations and derived data, and (potentially) slant values). The sub-sections below document the Fortran-90 derived types which map 1-to-1 with the contents of a *vfile* in the specification document, with the addition of internal quality control flags.

7.6.1 *Vfile-Header* parameters

The **COSTheader** derived type contains the *vfile-header* meta-data parameters.

```
TYPE COSTheader
   CHARACTER (LEN=4)      :: FmtVer          ! File format ident code (Vm.n)
   CHARACTER (LEN=20)     :: Project         ! Project name
   CHARACTER (LEN=20)     :: FileStatus      ! File status (OPER, DEMO or TEST)
   CHARACTER (LEN=4)      :: StnID           ! Station ident code
   CHARACTER (LEN=9)      :: DOMES           ! Station DOMES registration number
   CHARACTER (LEN=100)    :: FullName        ! Station full name
   CHARACTER (LEN=20)     :: RecType         ! Receiver type
   CHARACTER (LEN=20)     :: AntType         ! Antenna  type
   REAL(dp)               :: StnLat          ! Station latitude  wrt WGS-84 (deg)
   REAL(dp)               :: StnLon          ! Station longitude wrt WGS=84 (deg)
   REAL                   :: StnHt           ! Station height    wrt WGS-84    (m)
   REAL                   :: StnHMSL         ! Station height    wrt EGM96     (m)
   REAL                   :: StnHBMK         ! Station height above benchmark (m)
   CHARACTER (LEN=20)     :: DateTimeFirst   ! Date/time of first sample
                                               (dd-MMM-yyyy hh:mm:ss)
   CHARACTER (LEN=20)     :: DateTimeProc    ! Date/time of processing
                                               (dd-MMM-yyyy hh:mm:ss)
   CHARACTER (LEN=20)     :: ProcCentre      ! Processing centre name
   CHARACTER (LEN=20)     :: ProcMethod      ! Processing software type
   CHARACTER (LEN=20)     :: OrbitType       ! GNSS orbit source and type
   CHARACTER (LEN=20)     :: MetSource       ! Source of surface meteorological data
   CHARACTER (LEN=100)    :: ComSolLst       ! List of Processing Centre IDs (if a
                                               combined solution)
   INTEGER                :: TimeInc         ! Sample time increment/resolution (mins)
   INTEGER                :: UpdateInt       ! Batch update interval (minutes)
   INTEGER                :: BatchLen        ! Total length of batch data (minutes)
   INTEGER                :: PCDH            ! Header Product Confidence Data
   INTEGER                :: NumSamples      ! Number of data samples in file
 END TYPE
```

7.6.2 Single slant delay & associated parameters

The **`COSTslant`** derived type contains the *vfile-data* parameters for a single slant observation. The COST-format file may contain zero up to **`MaxSlants`** (currently 24) slant observations per ZTD timestamp.

```
  TYPE COSTslant
   CHARACTER (LEN=4)      :: SatID           ! GNSS satellite ident
   REAL                   :: STD             ! Slant Total Delay (mm)
   REAL                   :: STDErr          ! Estimated error in STD (mm)
   REAL                   :: Azim            ! Slant azimuth angle wrt North (deg)
   REAL                   :: Elev            ! Slant elevation wrt horizon (deg)
   INTEGER                :: QC              ! 0=OK, 1=FAIL
 END TYPE
```

7.6.3 *Vfile-Data* parameters

The **`COSTsample`** derived type contains the *vfile-data* parameters for a single timestamped ZTD sample observation. The COST-format file may contain zero up to **`MaxSamples`** (currently 96) samples per *vfile*.

```
  TYPE COSTsample
   INTEGER, DIMENSION(3) :: TimeStamp        ! Data sample time stamp (hr,min,sec)
   INTEGER               :: PCDD             ! Data Product Confidence data
```

```
    REAL                 :: ZTD              ! Zenith Total Delay (mm)
    REAL                 :: ZTDErr           ! Estimated error in ZTD (mm)
    REAL                 :: ZWD              ! Zenith Wet Delay (mm)
    REAL                 :: IWV              ! Integrated Water Vapour (mm)
    REAL                 :: SurfPres         ! Surface pressure (hPa)
    REAL                 :: SurfTemp         ! Surface temperature (K)
    REAL                 :: SurfRH           ! Surface relative humidity (%)
    REAL                 :: GradientN        ! N/S gradient (mm)
    REAL                 :: GradientE        ! E/W gradient (mm)
    REAL                 :: GrdErrN          ! Estimated error in N/S gradient (mm)
    REAL                 :: GrdErrE          ! Estimated error in E/W gradient (mm)
    REAL                 :: TEC              ! Total electron content (TECU)
    INTEGER              :: QC               ! 0=OK; 1=Invalid timestamp;
                                             ! 2=all obs values missing; 3=1+2
    INTEGER              :: NumSlants        ! Number of slants this data sample
    TYPE (COSTslant)     :: Slant(MaxSlants) ! Define up to MaxSlants slants/sample
  END TYPE
```

7.6.4 A whole *vfile*

The **`COSTvfile`** derived type represents a single, complete COST-format vfile – header plus data – together with a set of Q/C flag bits. The latter will be zero (all bits unset) until Q/C is performed with **`GBGP_QCVFile()`**. A COST-format file may contain any number (including zero) *vfiles* on disk.

```
   TYPE COSTvfile
    TYPE (COSTheader)    :: Header           ! Header type
    TYPE (COSTsample)    :: Sample(MaxSamples)! Samples
    INTEGER              :: QC               ! Overall Q/C flags
  END TYPE
```

8. Quality control

As described earlier, the COST-format file I/O routines **`GBGP_ReadCOST()`** and **`GBGP_WriteCOST()`** provide very little quality-control (Q/C), being limited to detecting disk read and write I/O errors (the former possibly due to technically mis-formatted file content) and ignoring data pre-flagged from failed Q/C tests. These routines do not test the validity of data parameters; this functionality is performed in the **`GBGP_QCVFile()`** routine. The Q/C tests can be broken into three parts:

1. validity checks on the *vfile-header* parameters
2. duplicate GNSS station ID detection (*vfile-header* meta-data cross-checks with the GB-GNSS database)
3. validity checks on the *vfile-data* parameters

These check are described in summary below, in (more-or-less) the order of testing, but refer to the code for the ultimate details. All parameters are checked in some way, but If certain critical parameters, such as time, station ID or 3-D location, are found to be invalid (so the observation would unusable) an error message will be issued and a Q/C flag set. Normally the data so flagged would be rejected from further processing.

8.1 Header checking

All header parameters are *technically* checked; for text (string) data, this may involve checking that one of the allowed values is given (as listed in [RD.4]); for numerical data, the test is for the value being within the allowed range or not. There is no attempt here to determine if a value has *scientific* validity. This may be termed *basic* or *passive* Q/C.

- Ensure all text parameters (except the GNSS station full name string) are upper case;
- Fixup known duplicate GNSS station IDs; this uses an external look-up table (**`GBGP_SIDFIX`**) mapping one of the duplicate IDs – which to change is based on the station latitude or longitude – to a unique ID. Check the station ID is 4 characters long and is wholly alpha-numeric. Flag an error if the ID is invalid and substitute '**`XXXX`**';
- Check for 'unknown' or 'missing' or blank text-based values (including non-standard strings such as a station name of '?' when the specification calls for 'Unknown') and substitute appropriate defaults according to [RD.4]. Parameters checked include the Project ID, Status and GNSS DOMES number;
- Check that the Analysis Centre ID is 3 or 4 characters long and wholly alpha-numeric (a padding underscore is valid for the 4th character when the AC ID is naturally 3 characters, such as '**`GFZ_`**');
- Ensure that the station full name is capitalized and contains a valid ISO-3166 English country name and 2-letter country code [RD.12] formatted according to [RD.4]. One sub-check is a fixup to some common non-ISO standard country names; this uses an external look-up table (**`GBGP_ISOFIX`**) to map the incorrect name to the ISO version. Country names as provided in COST-format files are sometimes
    - mistyped (e.g. 'Hongary' → 'Hungary' or 'French Guyana' → 'French Guiana')
    - political rather then the required geographical name[6] (e.g. 'England' → 'United Kingdom')
    - local (e.g. 'Tahiti' → 'French Polynesia')
    - non-English (e.g. 'Nederlands' → 'Netherlands')
- Fixup some receiver and/or antenna ID strings; there are often alternative strings supplied by different ACs for clearly the same piece of kit (for instance with/without spaces or an abbreviation of part of the name), so one is arbitrary chosen from an (internal) look-up table.
- Check that the 'first data' and processing date/time strings are valid in the sense that they are formatted according to [RD.4] and all elements are within their proper range – for instance not claiming to be 31 June. An invalid first data date/time is flagged as an error; an invalid or missing processing date/time is assigned the current UTC date/time.

[6] *For E-GVAP purposes, we require not the political 'owner' but the geographical territory; if a political or geographic entity has been assigned an ISO-3166-1 code, then that should be used. We have also adopted the ISO-3166 English short form of country names [RD.12]. Thus while the Falkland Islands are politically 'GB', they have their own ISO code 'FK'. Similarly there are many Pacific islands which are politically French, but it is their ISO code we require here. On the other hand, the Canary Islands do not have a separate ISO code, and remain Spanish ('ES').*

- Range-check the station's (ellipsoidal) latitude, longitude and height values. Longitude values are silently mapped to the range 0—360deg. Flag an error if any are invalid.
- Check the given station geoidal height (taken to be height above mean sea level). If missing or invalid, calculate it using the EGM-96 undulation interpolated to the station lat/lon location (if the 3-D location is valid). Requires the external file **`EGM96_GRID`**.
- Range-check the height above benchmark. Unfortunately, a value of zero is valid, but ACs sometimes code zero for 'Missing' (we assume, since other ACs provide a non-zero value for the same station).
- Range-check/mask the PCD. There is no check that the individual bits are self-consistent.

8.2 Meta-data checking

In addition to the above basic Q/C on the header parameters, they can also – optionally – be checked against an external database (see Section 9) of known stations and ACs. If:

- SQLite3 support was included at the GBGP package build/install time;
- database support is enabled at run-time **(`GBGP_SDB`** is not **`NONE`** and is set to a valid path, or is blank or not set at all in which case a default path is tried);
- the database is found and can be accessed (read-only is sufficient)
- the station ID being tested is found in the database (it is not a brand new station not yet added to the database),

then certain header parameters can also be cross-checked against the database values for that station. The purpose of this *database* or *active* Q/C is three-fold:

1. to fill in missing, invalid or unknown parameters in the *vfile-header* using meta-data provided by other ACs for that station ID (if available);

2. to cross-check that processing of a particular station by more than one AC results in consistent meta-data – for instance that the station's 3-D location offsets are small and within acceptable limits, or that the country name and/or ISO code is the same. If there is a significant difference (perhaps the lat/lon in the COST-format file is separated by 1km relative to the lat/lon in the database) but the station is clearly the same one, then the cause must be investigated and one of the ACs needs to correct their values. Sometimes several ACs may be processing a station, so usually if there is an odd one out, that AC needs to check their processing. Alternatively, it can be the case that a station (antenna) has moved a little or its location refined after initial processing by the same AC, and so it is the database which needs updating;

3. the detection of duplicate station IDs. Since there is no international registration procedure to reserve a unique IGS-style 4-character ID (though assigning a DOMES number comes closest to reserving the associated ID too), it is all too often the case that different ACs will provide data for a particular ID but nevertheless are for completely separate locations – perhaps even on different continents! Duplicated IDs are not a problem for pure NWP assimilation, when the given lat/lon for each individual ZTD observation can be used, but NWP systems also filter on the station ID (separately or together with the AC ID) and many have a bias correction scheme which depends on the station ID. Just passive monitoring of data flow (timeliness and availability metrics) and quality (ZTD observation model difference statistics) also relies on a unique station identifier. Clearly, then, duplicates are bad news and result in statistics from more than one site being mixed up.

   The database check will in these cases flag likely duplicates by testing pairs of:
   - the DOMES numbers (if both COST-format file and database have one);
   - the ISO 2-letter codes (after any fixup in the basic Q/C);
   - the horizontal (lat/lon) separation distance;
   - the station height (ellipsoid and geoid) differences;
   - the station full name string. This is uses a 'fuzzy match' algorithm such that small differences (such as capitalization and spelling variants) are ignored, but larger scores indicating quite different strings are deemed significant.

Each test is independent, and may result in an error condition (and message); the degree of difference is mapped to a probability value and these are summed to give an overall likelihood that the ID is a duplicate; the Q/C flag bit is set if this is above a defined threshold. The exception is the station name test; ACs (and even IGS & EUREF) cannot seem to agree on a station's proper name and it has proven impossible to get agreement on which is 'correct' (or 'preferred'). Therefore if this test gives a 'significantly different' result, it is flagged as a warning, not an error, and it is given only a low weight towards the probability total.

If a new duplicate ID is detected and confirmed, the AC would normally be contacted to see if the ID can be changed at source, or if that's out of the control of the AC, to be changed to something unique within their processing or at the point of writing their original COST-format file. If this is not possible either, then as a last resort, a suitable mapping can be added to the station fixup list (`GBGP_SIDFIX`). Which of the two stations should change, and which may keep its ID? This is somewhat arbitrary, but to keep an ID, consider in decreasing priority:

1. does either station have a DOMES number assigned (stations having distinct DOMES numbers but the same ID should never occur); favour the one with a DOMES number;
2. is the existing ID still in active use? If it has ceased, favour the new (active) one; if still active, change the newcomer.

The new ID for the station to be changed is equally arbitrary; as a rule of thumb, a ceased station ID could have the last character changed to '`X`'; otherwise change to '`1`' or some other digit or to the first letter of the country code – whatever gives a unique ID. Any proposed new ID (for a new station or to modify a duplicate as described above) can be checked by anyone using the browser-based tool noted in Section 9.3). Once a unique ID is given (upstream, or in **`GBGP_SIDFIX`**), that station can be added to the database and its entry checked when present in subsequent COST-format files.

8.3 Data checking

All numeric data parameters in the *vfile-data* section are *technically* checked; the test is for the value being within the allowed range or not. There is no attempt here to determine if a value has *scientific* validity. This may be termed *basic* or *passive* Q/C. There is no data section testing equivalent to the database meta-data checks performed for the header parameters.

Each parameter in each data sample is range-checked; if slant data is present, those parameters are also range-checked. The exception to the latter is a test that the GNSS transmitting satellite ID letter is one of the defined codes; '`X`' is substituted if not. In practice, no AC currently provides slant data in NRT COST-format files.

8.4 BUFR pre-check

As well as the Q/C described above, the BUFR encoding (see **`cost2bufr.f90`** and **`gbgpbufr.f90`**) performs its own range pre-check for numerical parameters to ensure that all values are representable within their BUFR bit-width after scaling, as defined for those Table B element descriptors. In general, the BUFR template is capable of encoding all valid values, but in some cases (such as ZTD), the total range (or upper value) may be a little more limited. This is also an independent back-stop in case upstream range-checking has not been performed (the application did not call **`GBGP_QCVFile()`**). Any out-of-BUFR-range values are set to the BUFR missing data flag value to avoid throwing errors in the kernel encoding library which would otherwise prevent encoding of that whole batch of data entirely.

9. The GB-GNSS database

9.1 Introduction

The *GB-GNSS database* is an SQLite relational (SQL)[7] database consisting of several tables which together provide many meta-data items about individual GNSS Stations, Analysis Centres and Reference Networks. Related tables provide supporting non-GNSS external meta-data such as ISO country codes and names and nearby WMO surface synoptic and upper-air (radiosonde) stations. In addition a *history* table tracks the first and last ZTD sample observation timestamp for every station-AC combination and an *update* table automatically updates a timestamp whenever any other table is changed (any record added, modified or deleted).

The SQLite3 database engine [RD.9] was chosen for it's simplicity & robustness and has scalability and functionality well within the requirements of this project. It has been well validated and continues to be developed as “free & open source”. The whole engine is in effect embedded within the application (via a shared library) and does not require any special access or privileges for a client to connect with a central server. The database consists of a single file which is easy to manage, dump, backup or copy elsewhere; many programming languages support SQLite natively and there is even a free-to-use source Fortran-90 interface to the C-based SQLite3 library which provides sufficient functionality for our needs.

Apart from the **`sqlite3`** command-line interface, which is part of the SQLite package, there are several third-party GUI interfaces for managing a database without needing to be an expert in SQL (and for simple record editing, no SQL knowledge is needed at all). We use *SQLiteManager* [RD.10] which needs the Mozilla **`xulrunner`** system (also used by the Firefox browser; *SQLiteManager* is also available as a Firefox plugin). The *SQLite Browser* GUI is an alternative tool [RD.11].

The database is optionally used as part of the Q/C by the pre-conversion and BUFR encoding tools (see Section 8.2), as shown in Illustration 3 (only those tables significant to the Q/C are shown). Normally the database is found at run-time via the environment variable (EV) **`GBGP_SDB`**. If this EV set blank or not set at all, the default path **`GBGP_DATA/gbgnss.sdb`** is tried (with **`GBGP_DATA`** in turn defaulting to **`~/data/gbgp`**). However if the EV

- is set to **`NONE`**, or
- the file path (as set in the EV or a default path if the EV is unset or set blank) is not found, or
- a connection cannot be made (for instance the user does not have read access rights)

then the database is not used (the application may issue an informational message to that effect) and only basic Q/C will be performed.

9.2 Database tables

Here we give a brief description of each of the tables in the database. Since it is not intended for general users to access the database directly, it is beyond the scope of this document to provide full details. Knowledgeable users may inspect the schema, table columns and data using the command-line **`sqlite3`** or one of the GUI tools mentioned above.

The GB-GNSS tables are shown schematically in Illustration 4 in which the left-to-right and top-to-bottom order reflects increasing significance of each table to the dynamic Q/C procedure. The primary key and purpose and content of each table is presented in Table 1. In the Static/Dynamic column, **S** denotes that the table is static (rarely, if ever changed, and by manual update) and **D** is dynamic (changed often, normally automatically from external sources, with occasional maintenance by manual updates).

As can be seen, the primary key name follows the pattern of the principle table name; these tables then contain keys referring to secondary tables which contain expanded information which may be common to

[7] *Structured Query Language: often taken to be synonymous with the term relational database (or RDB), but is strictly the interface language used to interact with such a database.*

many records in the primary tables. This use-case is an excellent illustration of using a *relational database* to extract almost any combination of information rather than using a flat file with duplicate meta-data, or several such files with bespoke connections, requiring inflexible explicit programming to extract information using only a limited number filter options.

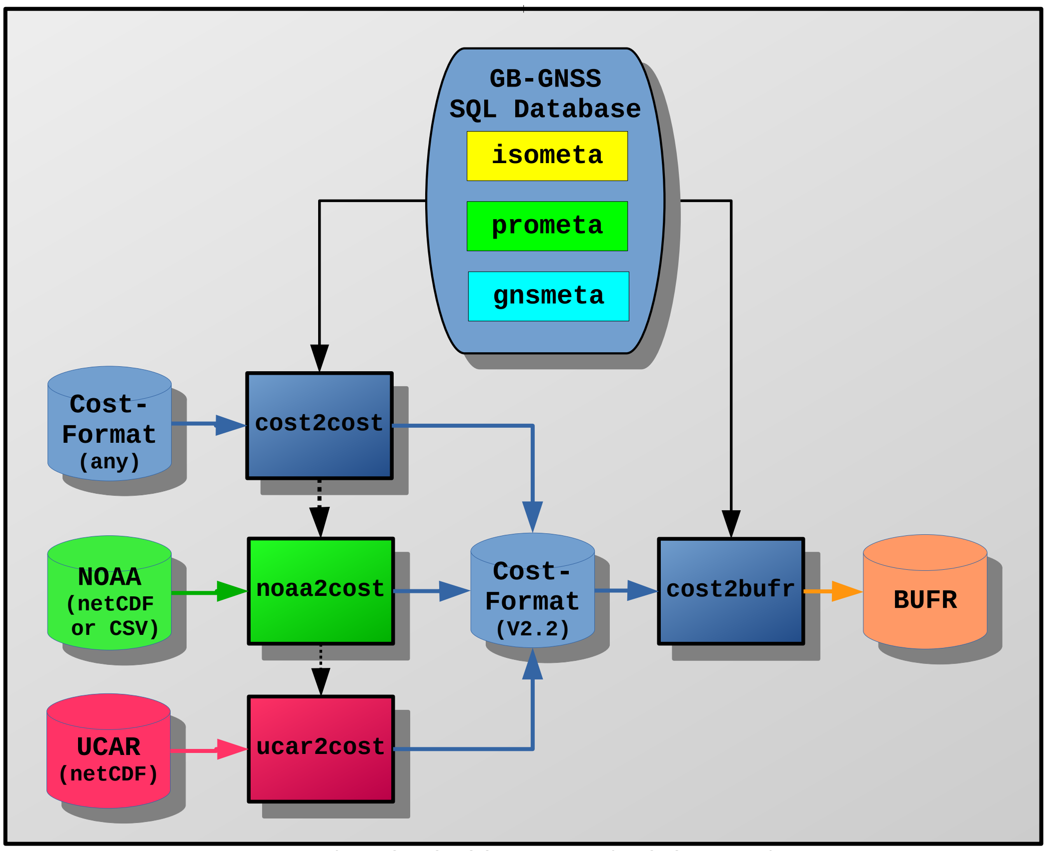


Illustration 3. The role of the GB-GNSS database for Q/C during format conversions

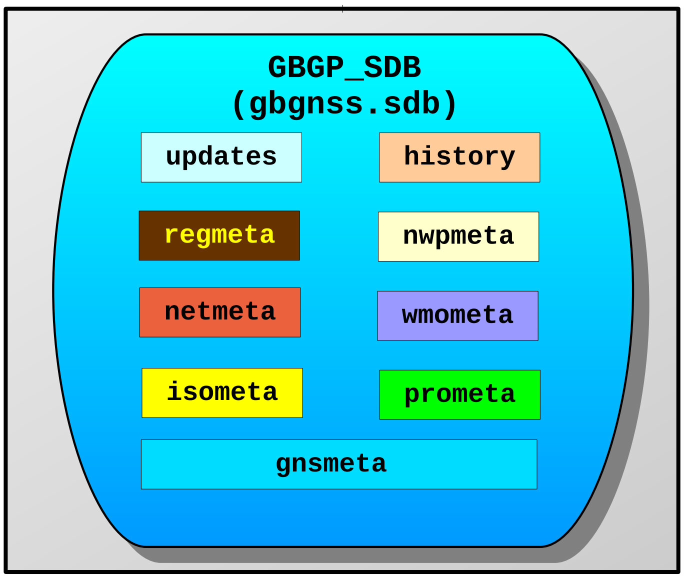


Illustration 4. GB-GNSS database tables

Table name	*Primary Key*	*Purpose, content, related table keys*	*Static or Dynamic*	*Principle Source*
gnsmeta	**gnsid**	GNSS station information, including: • meta-data specified in the COST-format *vfile-header* [RD.4] • ISO-3166-1 2-letter country code [key **isoid**] • Nearest WMO Surface (synoptic) and Upper-air (radiosonde) stations [key **wmoid**] • Reference Networks of which this station is part [key **netid**] • Processing (Analysis) Centre(s) currently processing this station [key **proid**] • Supersite flag (a station recommended to be processed by all ACs) • Comment (manual) and Updated (auto) text fields	**D**	NRT COST-format files
history	**combo**	Keeps track of the first and last observation (ZTD sample) timestamp provided for each combination of GNSS Station ID and Analysis Centre ID [keys **gnsid-proid**]. If the last timestamp is older than a defined threshold (e.g. 28 days), the AC is assumed to have stopped processing that station. A *ceased station* (no AC is processing it) or a *ceased AC* (the AC is no longer processing any stations) can similarly be deduced from this history.	**D**	NRT COST-format files
isometa	**isoid**	Provides a mapping between ISO-3166-1 2- or 3-letter or numeric codes and their ISO-3166-1 (English short) country names. Also includes alternative (variant) names to aid reverse look-ups.	**S**	[RD.12]
netmeta	**netid**	Reference Network information such as the full name, country code [key **isoid**] and website URL and a list of GNSS station IDs [key **gnsid**] in the network (whether actually in the **gnsmeta** table or not).	**D**	Various, including IGS, EUREF OSGB, SAPOS, etc[8]
nwpmeta	**nwpid**	NWP Centre information including: • Full name • Country code [key **isoid**] • Website URL • NWP model information	**S**	NWP centres
prometa	**proid**	Processing (Analysis) Centre information as contained in the COST-format *vfile-header* [RD.4], including: • Full name, country code [key **isoid**] and website URL • Region key (principle area of coverage) [key **regid**] • Current status flag (TEST, DEMO or OPER) • NWP model if used to derive data such as IWV [key **nwpid**]	**D**	NRT COST-format files
regmeta	**regid**	Region – full name and area of coverage defined by SW & NE latitude & longitude corners.	**S**	Local definition
updates	**tablename**	Timestamp for each table change (add, update, delete operation). Automatic updates defined by **TRIGGER**s attached to all other tables.	**D**	Automatic
wmometa	**wmoid**	WMO station information, including: • WIGOS ID • Full name and country code [key **isoid**] • Location (latitude, longitude, height AMSL) • Class and upper-air (radiosonde) flag	**D**	[RD.13]

Table 1.GB-GNSS database tables and their usage

9.3 Browser tool

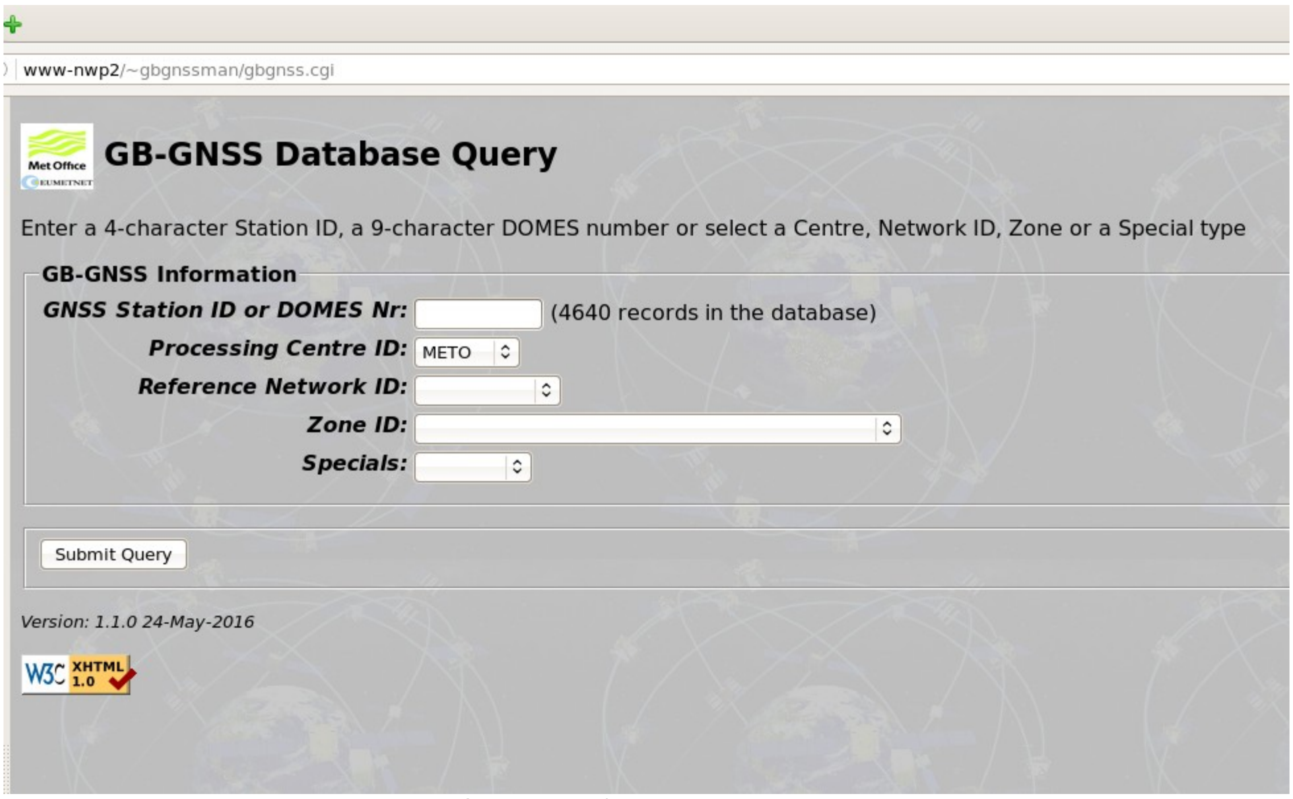
In order to provide a 'quick look' at the meta-data associated with a given GNSS station ID, or just to check whether a proposed new station ID is (or has ever been) in use – either provided in a COST-format file or is

[8] *The GFZ* ***semisys*** *database (https://semisys.gfz-potsdam.de/semisys/scripts/auth/login.php) contains an excellent centralised source for many networks, as long as site logs can be accessed. This database is used to keep the GBGP database up to date for those networks in semisys; some networks are not open and require a login account to access.*

in a known reference network – a CGI tool has been developed which can query the database and display the result in any modern web browser.

The tool can also query information on ACs, reference networks or stations in a geographical zone, and can filter stations by various special criteria such as *new* (added within the last 7 days) or *supersites* (a set of standard sites recommended to be processed by all E-GVAP ACs). A screenshot of the top-level query page is shown in Illustration 5; the user may type in any 4-character station ID or 9-character DOMES number, or select any entry from the drop-down lists for other types, in any combination. The query results will then be displayed – for example the output on selecting the processing centre ID **METO** is shown in Illustration 6. An individual station in the network list may be clicked on to display the associated meta-data, and the Station Maps and Station Listings links at the bottom of the result page will display (or download, depending on local browser settings) the network map (PNG) or text-based listings in long (LST) or short (CSV) format.

The Internet-facing and internal Met Office URLs are given at [RD.14]. The latter uses the live master database which is updated in near-real time with every uploaded COST-format file; the former is updated from a snapshot of the master database at least daily, using the script described in Section 6.2. Apart from this, the database and the browser tool are identical.


www-nwp2/~gbgnssman/gbgnss.cgi

GB-GNSS Database Query

Enter a 4-character Station ID, a 9-character DOMES number or select a Centre, Network ID, Zone or a Special type

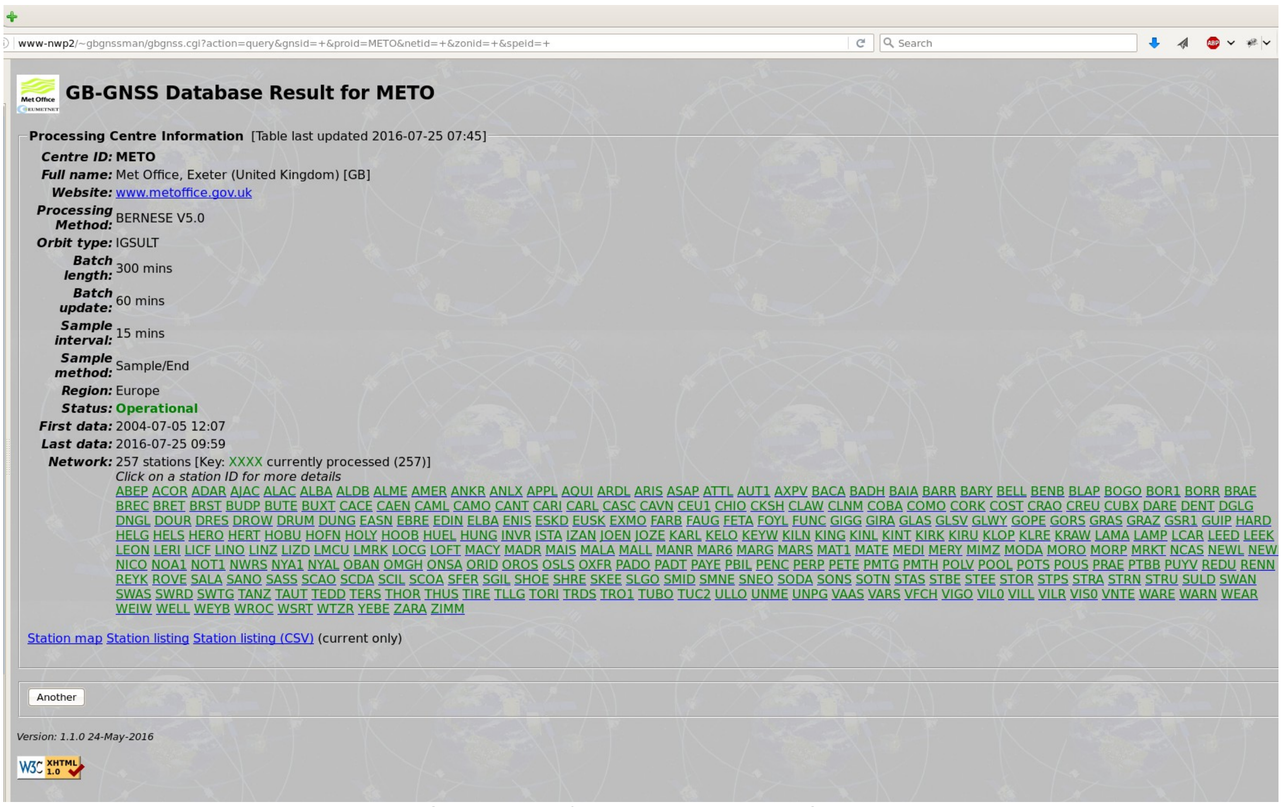
GB-GNSS Information

GNSS Station ID or DOMES Nr: (4640 records in the database)
Processing Centre ID: METO
Reference Network ID:
Zone ID:
Specials:

Submit Query

Version: 1.1.0 24-May-2016

Illustration 5. Screenshot of database browser tool query page


www-nwp2/~gbgnssman/gbgnss.cgi?action=query&gnsid=+&proid=METO&netid=+&zonid=+&speid=+

GB-GNSS Database Result for METO

Processing Centre Information [Table last updated 2016-07-25 07:45]

Centre ID:	**METO**
Full name:	Met Office, Exeter (United Kingdom) [GB]
Website:	www.metoffice.gov.uk
Processing Method:	BERNESE V5.0
Orbit type:	IGSULT
Batch length:	300 mins
Batch update:	60 mins
Sample interval:	15 mins
Sample method:	Sample/End
Region:	Europe
Status:	**Operational**
First data:	2004-07-05 12:07
Last data:	2016-07-25 09:59
Network:	257 stations [Key: XXXX currently processed (257)]

Click on a station ID for more details

ABEP ACOR ADAR AJAC ALAC ALBA ALDB ALME AMER ANKR ANLX APPL AQUI ARDL ARIS ASAP ATTL AUT1 AXPV BACA BADH BAIA BARR BARY BELL BENB BLAP BOGO BOR1 BORR BRAE
BREC BRET BRST BUDP BUTE BUXT CACE CAEN CAML CAMO CANT CARI CARL CASC CAVN CEU1 CHIO CKSH CLAW CLNM COBA COMO CORK COST CRAO CREU CUBX DARE DENT DGLG
DNGL DOUR DRES DROW DRUM DUNG EASN EBRE EDIN ELBA ENIS ESKD EUSK EXMO FARB FAUG FETA FOYL FUNC GIGG GIRA GLAS GLSV GLWY GOPE GORS GRAS GRAZ GSR1 GUIP HARD
HELG HELS HERO HERT HOBU HOFN HOLY HOOB HUEL HUNG INVR ISTA IZAN JOEN JOZE KARL KELO KEYW KILN KING KINL KINT KIRK KIRU KLOP KLRE KRAW LAMA LAMP LCAR LEED LEEK
LEON LERI LICF LINO LINZ LIZD LMCU LMRK LOCG LOFT MACY MADR MAIS MALA MALL MANR MAR6 MARG MARS MAT1 MATE MEDI MERY MIMZ MODA MORO MORP MRKT NCAS NEWL NEW
NICO NOA1 NOT1 NWRS NYA1 NYAL OBAN OMGH ONSA ORID OROS OSLS OXFR PADO PADT PAYE PBIL PENC PERP PETE PMTG PMTH POLV POOL POTS POUS PRAE PTBB PUYV REDU RENN
REYK ROVE SALA SANO SASS SCAO SCDA SCIL SCOA SFER SGIL SHOE SHRE SKEE SLGO SMID SMNE SNEO SODA SONS SOTN STAS STBE STEE STOR STPS STRA STRN STRU SULD SWAN
SWAS SWRD SWTG TANZ TAUT TEDD TERS THOR THUS TIRE TLLG TORI TRDS TRO1 TUBO TUC2 ULLO UNME UNPG VAAS VARS VFCH VIGO VIL0 VILL VILR VIS0 VNTE WARE WARN WEAR
WEIW WELL WEYB WROC WSRT WTZR YEBE ZARA ZIMM

Station map Station listing Station listing (CSV) (current only)

Version: 1.1.0 24-May-2016

Illustration 6. Screenshot of example Processing Centre result page