The EUMETSAT Network of Satellite Application Facilities

ROM SAF

Radio Occultation Meteorology

ROM SAF CDOP-2

GBGP Release Notes

Version 1.0 Prototype

1 June 2016

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

GBGP

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

DOCUMENT AUTHOR TABLE

	Author(s)	Function	Date	Comment
Prepared by:	D Offiler	GBGP Development Team Leader	1/6/2016	
Reviewed by:	O Lewis I Culverwell	GBGP Development Team ROM SAF Package Manager	1/6/2016	
Approved by:	K B Lauritsen	ROM SAF Project manager	1/6/2016	

DOCUMENT CHANGE RECORD

Issue/Revision	Date	By	Description
1.0-proto	1/6/2016	DO	1st prototype release version

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

Illustration Index

Index of Tables

Executive Summary

This document outlines the first ***prototype*** release of the Ground-Based GNSS Package (GBGP) Version 1.0-proto or just **GBGP-1**. It describes the file set, documentation, dependency packages, a summary 'Quick Start' guide to build/installation, a list of supported platforms and known problems.

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 use of the GBGP library functions and the executable tools in the package, please refer to the GBGP User Guide [RD.2].

This release is designated 'prototype', because the software has not yet undergone the ROM SAF's formal release procedure which includes code review, a comprehensive test suite, independent beta-testing, a Delivery Readiness Inspection Review and formal approval to release by the ROM SAF Steering Group. Nevertheless, the application code is effectively the same as that in the E-GVAP package which has been in regular use for many years. The refactored GBGP package has been informally tested, is known to be highly portable and is being used in the Met Office for operational dissemination of E-GVAP GB-GNSS data in BUFR via the GTS to global NWP users.

1. Introduction

1.1 Purpose of the document

This document outlines the first major release of the Ground-Based GNSS Package (GBGP) Version 1.0 or just **GBGP-1**. It describes the file set, documentation, dependency packages, a summary 'Quick Start' guide to build/installation, a list of supported platforms and known problems.

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

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

[RD.4] GBGP Test Plan
Ref: SAF/ROM/METO/TP/GBGP/001

[RD.5] 'COST-format' file specification for ground-based GNSS delay and water vapour data
Ref: E-GVAP/METO/FMT/COST/001

[RD.6] WMO FM94 (BUFR) specification for ground-based GNSS delay and water vapour data
Ref: E-GVAP/METO/FMT/BUFR/001

[RD.7] NetCDF (Unidata) website
URL: http://www.unidata.ucar.edu/software/netcdf/

[RD.8] NOAA/Earth Systems Resreach Laboratory GPSNet website
URL: http://gpsmet.noaa.gov/

[RD.9] UCAR/COSMIC Programme Office SuomiNet/CONUS website
http://www.suominet.ucar.edu/

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[1]
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	Euopean 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
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)

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

ROM SAF	Radio Occultation Meteorology SAF (formerly GRAS SAF)
GBGP	Radio Occultation Processing 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
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 the E-GVAP project and other data suppliers (such as NOAA and 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, metadata 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, metadata, 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.5];
BUFR:	WMO binary format for dissemination of NRT observational data on the GTS. For GB-GNSS details, see [RD.6];
netCDF:	Unidata binary format for general data storage and exchange [RD.7]. For GB-GNSS data and documentation on this format, see [RD.8] and [RD.9];
CSV:	A simple text-line-based flat-format. For GB-GNSS data and documentation on this format, see [RD.8].

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 prototype release of the Ground-Based GNSS Package (GBGP) Version 1.0-proto or just **GBGP-1**. It describes the file set, documentation, dependency packages, a summary 'Quick Start' guide to build/installation, a list of supported platforms and known problems.

The GBGP is a collection of software (provided as source code in Fortran-95), 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.

Section 6 has guidance for unpacking and building GBGP. For details on the use of the GBGP library functions and the executable tools in the package, please refer to the GBGP User Guide [RD.2].

The download website is organized as:

```
GBGP Download (root) page
  |-- Documents
  |-- Dependency Packages
  |-- Previous Releases
```

The root directory contains the Release Notes (this file), a Change Log [RD.1] and compressed tar archive files ('tarball') which provide the latest complete release for GBGP.

Related documentation included within the tarball is also available separately in the **`Documents`** sub-directory.

Third-party dependency packages (not part of the GBGP distribution and for those whose licences permit it) can be downloaded from the **`Dependency Packages`** directory.

The **`Previous Releases`** directory contains a small number of older GBGP releases (when there are any[2]), but please note that support for these is limited and decreases with age.

Section 3 gives full details of the files contained in this download area.

GBGP pre-conversion and encoder tools have been fully validated using NRT GB-GNSS data from several sources including E-GVAP suppliers [RD.4], NOAA [RD.7] and UCAR [RD.8]. As such, it is guaranteed for operational use.

All of the files described here can be downloaded by registered and logged-in users from the ROM SAF website:

`http://www.romsaf.org > User Login > GBGP > GBGP Download`

Before being able to download any GBGP files, you will have accepted the Terms and Conditions of the full GBGP Software Licence, a copy of which is available in the **`Documents`** directory for reference. Note also the contents of the **`COPYRIGHT`** file in the tarball.

All comments on, or questions about, the GBGP software should in the first instance be reported via the ROM SAF Helpdesk at **`http://www.romsaf.org > Helpdesk`**

[2]*Since this is the first full release of GBGP, there are currently no previous versions.*

3. Release file set

The 'Status' column in the following tables shows the current status of files in this package (v1.0-proto) since the previous development release[3] (v1.0-devel), by a traffic light system:

Unchanged **Updated** **New**

A summary of the main changes to this package from the previous release are documented in the Change Log [RD.1].

3.1 GBGP root directory

Table 1 Shows the files in the top-level GBGP Downloads directory.

File Name	*Description*	*Version*	*Status*
`GBGP_Release_Notes.pdf`	This file	V1.0	**New**
`GBGP_Changes.pdf`	Change Log - summary of changes from Release Version 1.0-proto to 1.0 [RD.1]	v1.0	**New**
`gbgp-1.0.0-proto.tar.gz`	Compressed tar archive (tarball) of the whole GBGP-1 (V1.0.0-proto) distribution set	V1.0.0-proto	**New**
`gbgp_buildpack-1.0.0-proto.tar.gz`	Compressed tar archive (tarball) of the GBGP-1 (V1.0.0-proto) build support – see Section 6.4.	V1.0.0-proto	**New**

Table 1. Files in the root directory

3.2 Documents directory

The GBGP User Guide and documentation on GBGP code (Reference Manual), file formats (COST-format and BUFR), Technical Notes and other related items, can be found here. Table 2 lists the files in this directory. All are contained in the main package tarball but are provided individually here.

File Name	*Description*	*Version*	*Status*
`README.docs`	Additional notes on the documents	V1.0	**New**
`romsaf_gbgp_licence.pdf`	GBGP User Licence	V1.0	**New**
`GBGP_User_Guide.pdf`	GBGP User Guide. [RD.2]	V1.0	**New**
`GBGP_Reference_Manual.pdf`	GBGP Reference Manual. [RD.3]	V1.0	**New**
`egvap_cost_v22a.pdf`	E-GVAP COST-format specification for GB-GNSS data [RD.5]	V2.2a	**Unchanged**
`egvap_bufr_v13.pdf`	E-GVAP BUFR template specification for GB-GNSS data [RD.6]	V1.3	**Unchanged**

Table 2. Files in the documents directory

[3] *For internal ROM SAF testing only; not a general public release.*

Please refer to the User Guide [RD.2] for details on the usage of the package. For users wishing to write their own interfaces to the GBGP routines, or to modify the GBGP code, detailed user-documentation in the form of a Reference Manual is provided [RD.3]. The Reference Manual is also contained in the main GBGP tarball. Stand-alone application tools also have Unix-style 'man' pages which are installed during the GBGP building process.

3.3 Dependency Packages directory

GBGP uses some standard third-party packages. Where licences permit, the latest versions known to be compatible with GBGP can be downloaded from the GBGP Download page on the ROM SAF website; these are listed in Table 3. The file **`README.deps`** documents the original source of these packages. See Section 6 for installation tips.

File Name	*Description / Source website*	*Version*	*Status (see Note)*
`README.deps`	Additional notes on the dependency packages	V1.0	**Unchanged**
`bufr-24.0.2.tar.gz`	Met Office BUFR kernel library and generic tools http://www.romsaf.org	v24.0.2	**Unchanged**
`sqlite-autoconf-313000.tar.gz`	SQLite3 library (optional for metadata checking) http://www.sqlite.org/	V3.13.0	**Unchanged**
`hdf5-1.8.16.tar.gz`	HDF5 software (required for netCDF-4). http://www.hdfgroup.org/HDF5/	v1.8.16	**Unchanged**
`netcdf-c-4.4.0.tar.gz`	NetCDF-4 core library http://www.unidata.ucar.edu/software/netcdf/	V4.4.0	**Unchanged**
`netcdf-fortran-4.4.3.tar.gz`	NetCDF-4 Fortran interface library http://www.unidata.ucar.edu/software/netcdf/	V4.4.3	**Unchanged**
`zlib-1.2.8.tar.gz`	ZLIB software (optional for HDF5). http://www.zlib.net/	v1.2.8	**Unchanged**

Table 3. Files in the dependency packages directory

Notes:

1. Later versions may become available from the referenced source websites; we do not guarantee that these will work with GBGP. If later versions are proved to be compatible, the new distribution files will be posted here.
2. For **`BUFR`** encoding, only the Met Office 'MetDB' library is supported by the GBGP system. It is only required for the primary BUFR encoder tool; if the library is not found, this application will not be compiled.
3. The **`SQLite3`** library is used for quality-control using a relational database of station and processing centre meta-data. Where possible we recommend that a system-installed package is provided via your distribution (distro) package manager or equivalent pre-built binaries. Any version release V3.6.0 or later should be acceptable. If the library is not found, the GBGP tools will be built without this functionality, but only basic quality-control (such as parameter validity range checking) will be performed.
4. **`NetCDF-4`** is the current netCDF release family; while GBGP currently uses only the 'classic' netCDF model (equivalent to legacy **`netCDF-3`**) the build system only supports **`netCDF-4`** (core release V4.4.0 or later), in which the core C library and Fortran codes are provided in separate tarballs). NetCDF is not required for the principle BUFR encoder application, only for the pre-converter tools; if the dependency is not found when building GBGP, a warning will be issued and these tools will not be compiled.
5. The **`HDF5`** software suite is only required if **`netCDF-4`** capability is also needed to build the GBGP pre-converter tools. Where possible we recommend that a system-installed **`HDF5`** is provided via your distribution (distro)

package manager or equivalent pre-built binaries. Any version release V1.8.9 or later should be acceptable. **`HDF5`** should only be built locally by the non-privileged user if a system **`HDF5`** is not available and it is not possible to install one.

6. The **`ZLIB`** library is only needed if **`HDF5`** compression is desired. Since almost all Linux/Unix systems come with **`ZLIB`** as standard, most users will not have to download and build the package for themselves. The **`HDF5`** configure should find the system **`ZLIB`** automatically, but it will build without compression support if it cannot be found.

Build and configuration scripts are included in the GBGP tarball to assist in building these packages consistently for use with GBGP; see Section 6.4. These files are also provided in a separate **`buildpack`** tarball as a convenience to the user.

4. Previous versions

If you have installed any previous version of GBGP we recommend that you install GBGP-1 to a separate target path – do not just install over an existing installation. This is especially important if the earlier tools or applications built on those libraries are being used for operational production. Only swing over to using the new GBGP-1 tools and libraries when you have validated them locally and you are satisfied that they meet your requirements.

Support for any previous full releases[4] will be limited to problems which can be reproduced with the latest release of GBGP-1. If reported problems have been already fixed at this version, we recommend upgrading to the latest.

All previous Prototype and Beta release versions of ROPP, up to and including v1.0-beta, are obsolete and are completely unsupported. If you have previously installed any version of GBGP for Beta testing, under the terms of the Beta licence (to which you agreed when first requesting access to the software) the right to use this software terminated after one year. Likewise, the Prototype (informal) Release package license expires after a maximum of two years and is superseded by a later v1.0 (full) latest release. Any and all GBGP Prototype and Beta files downloaded for testing and evaluation purposes are unsupported and should be deleted after implementing the full package.

[4]*As this is the first prototype release of GBGP (discounting internal development versions), there are currently no previous releases to support and there are no tarballs in the* **`Previous Releases/`** *download directory.*

5. Quick-start Guide

This section provided a brief overview to building and installing the GBGP software. We shall assume a default setup in that:

1. all necessary dependencies are pre-installed (see Section 6.2)
2. the GNU **`gcc`** C-compiler and Intel **`ifort`** Fortran 95 compilers are installed
3. the dependency libraries (shared or static) were built using the same compilers

For details on how to use different compilers see Section 6.4. Proceed as follows:

1. create a suitable directory or chose an existing one (for instance **`~/Downloads`** or **`~/packages`**)
2. **`cd`** to this directory and download or copy the main GBGP distro and buildpack tarballs here
3. unpack the buildpack tarball:
   ```
   > tar -zxvf gbgp_buildpack-1.0.0-proto.tar.gz
   ```
4. set the installation target; for example:
   ```
   > export GBGP_ROOT=$HOME
   ```
   if environment variable **`GBGP_ROOT`** is undefined, it will default to this package directory.
5. Build & install GBGP:
   ```
   > ./gbuildpack gbgp local
   ```
   or use the wrapper script:
   ```
   > ./gbuild_gbgp
   ```
   which will unpack the GBGP tarball, perform extra checks and save a logfile of the process using all default settings.

6. Unpacking and building GBGP

This section provides guidance on building the complete GBGP package as a standalone installation for evaluation and testing or to build the application tools for local use. GBGP is designed to be highly modular and key components of the software may be extracted for embedding in user applications via user-callable application program interface (API). In the latter case, some or all of the noted 3rd party dependency packages may not be required. See the User Guide [RD.2] for guidance on use of the application tools and main GBGP APIs.

6.1 Unpacking

The file **gbgp-1.0.0-proto.tar.gz** is a compressed tar archive (approx. 5Mb) which contains the complete GBGP-1 (v1.0.0-proto) distribution (distro) file set. Download (or copy) this file to a suitable *package directory* (for example **~/Downloads** or **~/packages**), where the content can be extracted with:

```
> tar -zxvf gbgp-1.0.0-proto.tar.gz
```

or

```
> gunzip -c gbgp-1.0.0-proto.tar.gz | tar -xv
```

if your version of **tar** does not support decompression (**-z** option).

This will create a sub-directory with the following structure:

```
gbgp-1.0.0-proto/
     |- build/              intermediate build files
     |- buildpack/          GBGP and dependency package build support scripts
        |- configure/       package configuration ‘mini-scripts’
     |- compilers/          compiler-specific source code
     |- data/               reference and run-time files
     |- doc/                Reference Manual
     |- gbgp/               GBGP library source code
     |- man/                ‘man’ pages for GBGP application tools
     |- scripts/            shell scripts
     |- tests/              user-tests
     |- tools/              GBGP application tools source code
     |- utils/              generic utility source code (‘pre-existing software’)
```

The **gbgp-1.0.0-proto/** top (package root) directory also contains additional **README** files and various other text files.

The second file **gbgp_buildpack-1.0.0-proto.tar.gz** is a compressed tar archive (approx. 25Kb) which contains the same files that are in the **buildpack/** directory of the main distro tarball, but in a simpler form to extract to the user’s package directory. For instance, within the same package directory, you can:

```
> tar -zxvf gbgp-1.0.0-proto.tar.gz
```

or

```
> cp -R gbgp-1.0.0-proto/buildpack/* ./
```

to prepare the supporting package-specific build (wrapper) scripts **gbuild_<package>**, the generic **gbuildpack** shell script and the configure mini-scripts.

We suggest that any required dependency package archive files (Section 6.2) are copied or downloaded to this package directory; they can then be unpacked and built using the **gbuildpack** tool (Section 6.6) or the appropriate wrapper scripts (Section 6.7), *before* building the GBGP module itself.

6.2 Dependency packages

Third-party dependency libraries must be installed before attempting to build any GBGP code. The **`configure/`** sub-directory contains some configuration mini-scripts (see Section 6.4) and there are higher-level shell scripts **`gbuildpack`** (Section 6.6) and **`gbuild_<package>`** (Section 6.7) to assist in building the **`ZLIB`**, **`HDF5`**, **`netCDF-4`**, **`SQLite3`** and **`BUFR`** packages. Review and edit these to suit your local requirements before running them. *However, we recommend that system-installed **`ZLIB`**, **`HDF5`** and **`SQLite3`** libraries (together with their development header files) be used where possible.*

Illustration 1 shows an overview of the GBGP application tools and the dependency packages used by each. All of these dependency packages are in a sense optional, in that if not found when building GBGP, those components or tools which require the package will lack some functionality or not be built at all (and only a warning will be issued), but other components/tools with satisfied dependencies will still be built. Table 4 shows for which GBGP component(s) each dependency package is used, and what the build system will do if that package is not found when making the GBGP library and application tools.

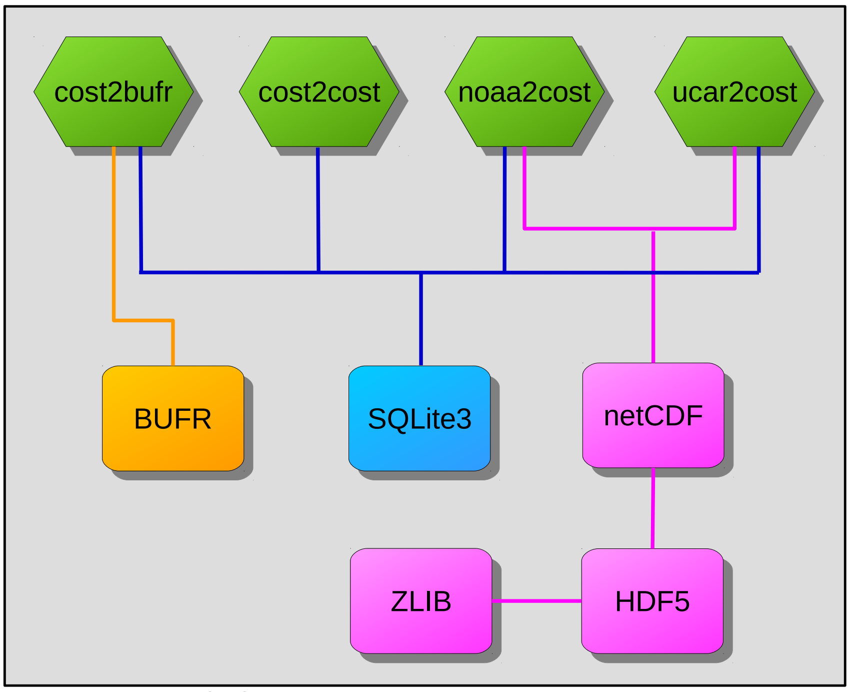

Illustration 1. GBGP application tools and their dependency packages

Any third-party packages already installed (e.g. for previous GBGP builds or for other applications) do not normally need to be re-built but should be in the expected GBGP build path for libraries, as real files or as symbolic links or in standard system paths. Note that this release may require a minimum version of a dependency package. Third-party packages may have been updated since the previous release of GBGP; those available on the ROM SAF GBGP download website in the **`Dependency Packages/`** directory are the latest versions supported by GBGP and the package build scripts. We recommend installing these latest supported versions, but in general it is not necessary to do so. Rebuilding is also recommended if your C or Fortran compiler version has changed since the original build.

Dependency package	***Used by***		***If not present***
`BUFR`	BUFR encoder	**`cost2bufr`**	Application not built
`SQLite3`	GBGP library BUFR encoder NOAA pre-converter UCAR pre-converter	**`libgbgp.a`** **`cost2bufr`** **`noaa2cost`** **`ucarcost`**	Dummy Fortran SQLite interface substituted in GBGP library Applications will not perform run-time station metadata checks against an SQL database
`NetCDF-4`	GBGP library NOAA pre-converter UCAR pre-converter	**`libgbgp.a`** **`noaa2cost`** **`ucarcost`**	**`NetCDF-4`** utility support not included in GBGP library Applications not built
`HDF5`	**`NetCDF-4`** core C-library		**`NetCDF-4`** (core and Fortran) packages not built
`ZLIB`	**`HDF5`**		**`HDF5`** built without compression functionality

Table 4: GBGP dependency package usage and consequences if not present

It is highly recommended that third-party dependency packages are compiled using the same C and Fortran compilers being used for the GBGP code. Mixing code at object level, even using different versions of compilers from the same vendor, can cause linking or run-time problems. C code in the dependency packages must be compiled using the **`cfortran.h`** system or equivalent to ensure calling interface compatibility with the Fortran compiler's object code, which means that the package must be configured for use with a particular Fortran interface, even if the package itself contains no Fortran (this does not apply to **`SQLite3`**, **`ZLIB`** or **`HDF5`** which are not accessed directly from Fortran, which is why system-installed versions can – and should –- be used where available).

6.3 Environment variables

The build system uses an environment variable **`GBGP_ROOT`** as the root path for all file installations. By default, this will be set to **`$HOME`** by the **`gbuildpack`** script. Pre-define this variable before installing any 3rd party packages or GBGP; for instance:

```
> export GBGP_ROOT=/usr/local
```

noting that the user must have appropriate file permissions to write to such a location. Typically, a working installation will be to the user's home space, so:

```
> export GBGP_ROOT=$HOME
```

The GBGP BUFR encoder uses the Met Office 'MetDB' BUFR kernel library (if not present the application will not be built) which requires an environment variable to be pre-set to the directory path for the run-time tables. For instance:

```
> export BUFR_LIBRARY=$GBGP_ROOT/data/bufr
```

Please refer to the documentation that comes with with the BUFR package on how to install this library and related files or simply use the **`gbuildpack`** script with a **`<package>`** name of **`mobufr`** or just **`bufr`** (or preferably use the **`gbuild_bufr`** wrapper script). This environment variable can be pre-defined so that **`gbuildpack`** will install the BUFR run-time files to the desired location (the above examples shows the default path). Building GBGP will also install BUFR-related files to this location.

Other environment variables are used to define paths and to control the behaviour of the GBGP application tools at run-time; refer to the GBGP User Guide [RD.2] for details of these.

6.4 Configuration mini-scripts

The **`configure/`** sub-directory contains example shell 'mini-scripts' for configuring the build and installation of not only the GBGP code but also the dependent third-party packages in a compatible way. These files follow the naming style **`<package>_configure_<compiler>_<os>`** where **`<package>`** is the package name (**`gbgp`**, **`netcdf`**, **`bufr`**...), **`<compiler>`** is the compiler ID (**`ifort`**, **`nagfor`**, **`pgf95`**, **`gfortran`**, **`ftn`**...) and **`<os>`** is the operating system ID, as output by the **`uname`** command but entirely in lower case (**`linux`**, **`cygwin`**,...). For instance, the mini-script file **`netcdf_configure_gfortran_linux`** can be used to build the core (C-only) **`netCDF-4`** package for use with GFortran on a Linux system.

There is also a set of mini-scripts named **`<package>_configure_local`**; when any of the build scripts are given the 'compiler' ID **`local`** these versions are used whatever the OS. As provided, these mini-scripts configure for compiler **`ifort`** on Linux, but the user is encouraged to edit them to use whatever local default compiler is desired.

These configure mini-scripts are used by the utility **`gbuildpack`** script or can be used for manual configurations – see Section 6.8.

The example configure mini-scripts for specific platforms and compilers may need to be edited for optimal local use, or users may create their own following one of the examples. We would be pleased to receive back equivalent scripts if you have built the packages using different compilers and/or platforms and which could be included in future updates. We can't formally support such scripts since their use is highly dependent on the user's target system and compiler setting preferences and we have no means of testing them ourselves.

6.5 Installation tree

The GBGP installer will copy working files to the standard tree at root *prefix* **`GBGP_ROOT/<compiler>`** (or, when *compiler* is **`local`**, *prefix* is just **`GBGP_ROOT`**)

`<prefix>/bin`	binaries (executable programs and shell scripts **`*.sh`**)
`<prefix>/lib[64]`	object libraries (static **`*.a`** and shared **`*.so`**)
`<prefix>/include`	include files (**`*.h`**, **`*.mod`**)
`<prefix>/share`	man pages, etc
`<prefix>/data`	run-time data files

As this is a typical tree for many packages, these directories may already exist when *prefix* is **`HOME`** and *compiler* is **`local`**.

6.6 The gbuildpack script

Included in the package is a Bash shell script file **`gbuildpack`**. The user need not be using Bash as an interactive shell, but Bash needs to be installed (usually at **`/bin/bash`**) for the script to work; almost all POSIX-based systems should have Bash installed by default, and for many Linux distributions (distros), Bash is the default login shell. On the rare systems that do not have Bash installed, follow the main commands for the desired package installation section of this script or follow the manual build procedures documented for each package (summarized in Section 6.8) – except for BUFR, all follow the common **`configure`** / **`make`** / **`make install`** method.

This script may be used to automate the build of GBGP or any dependency package in a consistent way, using the above mentioned configure mini-scripts. Summary usage can be obtained by:

```
> ./gbuildpack -h
```

This script assumes that all archive files (tarballs) are placed in the same working directory and the configure mini-scripts are in a **`configure/`** sub-directory. Packages will be extracted here and resulting files installed to the **<prefix>** target directory. In general, to build and install a package, it is only necessary to give the command:

```
> ./gbuildpack <package> <compiler> [[NO]CLEAN]
```

where **<package>** is one of the supported GBGP or dependency package names (**gbgp**, **bufr**, **sqlite**, **netcdf**, **netcdff**, **hdf5** or **zlib**) and **<compiler>** is the required compiler (**ifort**, **nagfor**, **gfortran**, **pgf95**, **local**, ...) which matches an appropriate configure script. The **<os>** part of the file name is detected automatically.

The script's main steps are:

1. perform various checks on the environment, install target and available packages;
2. optionally delete the package source tree to force a clean build (if **CLEAN** is present);
3. if no source tree is present (first install or deleted at step 2), unpack the latest archive file (tarball) or:
4. clean a pre-existing source directory (optionally skip this step if **NOCLEAN** is present);
5. run the appropriate mini-configure script, build, test and install commands.

Since some modules are dependent on other modules, the recommended build order is:

1. If requiring the GBGP pre-conversion tools:
   - i. **zlib** but only if ZLIB is not system-installed
   - ii. **hdf5** but only if HDF5 is not system-installed
   - iii. **netcdf** the core C-only package
   - iv. **netcdff** the Fortran API
2. **bufr** if the GBGP BUFR encoder tool is required
3. **sqlite** but only if SQLite3 is not system-installed and SQL database checking is required
4. **gbgp** the GBGP application package

Note that this Bash script has only been tested under Linux (RHEL6, OpenSUSE 13 and Cray LE), and Cygwin (under Windows 7 & 10) and may need minor modifications to run on other systems. *Users are strongly advised to build the external dependency packages with* **gbuildpack** *or the wrapper scripts described in the next section.*

6.7 Wrapper scripts

Shell wrapper scripts **gbuild_<package>** are provided which can be used to further automate the build process by calling **gbuildpack** with a predetermined (or given) sequence of compilers for a given package and which save a copy of all screen output to a disk log file. In general, run the command:

```
> ./gbuild_<package> -h
> ./gbuild_<package> [compiler …] [option]
```

For instance:

```
> ./gbuild_sqlite gfortran pgf95
```

will build the SQLite3 package using the **gfortran** and **pgf95** configurations. If no compilers are given on the command line, the default of **local** is used; this is the compiler defined in the **<package>_configure_local** mini-script. if the option is **all**, then *all* of the compilers used in our development testing environment will be used in turn (if installed).

Note that **gbuild_netcdf** will (by default) build both the **netcdf** (core) and **netcdff** (Fortran) packages. Use the **-noc** option to skip the core build or **-nof** to skip the Fortran build (but of course only when the core package has already been built).

The final script **gbuild_deps** will build a predetermined (or given) set of packages for a given compiler, thus:

```
> ./gbuild_deps <compiler> [package …]
```

for example:

```
> ./gbuild_deps ifort netcdf
```

Review and edit these scripts – for instance the default list of Fortran compilers or dependency packages – to suit your requirements.

For a final working built with your preferred default compiler

1. consistently edit the set of mini-scripts ***_configure_local**
2. set **GBGP_ROOT** to **HOME** (or wherever)
3. run the appropriate build script with (default) **local** as the compiler ID.

Assuming the wrapper script completes without detecting any errors, it will show the path to the log file for each run of **gbuildpack**; review these logs to confirm that the build(s) were indeed successful. If the wrapper script detects an error reported by **gbuildpack**, it will stop immediately.

Less experienced users are strongly encouraged to use these wrapper scripts; should problems arise which cannot be resolved by the user, the log file(s) can be posted to the ROM SAF Helpdesk for support.

6.8 Manual package building

The GBGP and dependency packages (apart from the MetDB BUFR package) all employ the common **autotools**-based build and install system. The generic sequence is:

```
> tar -zxvf <package>-<ver>.tar.gz
> cd <package>-<ver>
> [environment settings..] ./configure [options...]
> make
> make check
> make install
```

One of the mini-scripts for the package in **GBGP-1.0/configure** can be used as-is or modified for local use in the above **configure** line. The environment and target install directory settings of course must be consistent across all packages for a correct build and run-time of the final GBGP applications.

The BUFR package doesn't use **autotools** but instead a custom build/install script which drives pre-written **Makefile**s:

```
> tar -zxvf bufr-<ver>.tar.gz
> cd bufr-<ver>
> ./buildbufr [options...]
```

Run **buildbufr -h** for a list of available options. The most common are:

-c	clean the source tree before re-building (for instance with a different compiler)
-f	Fortran compiler (default **ifort**)
-p	prefix target directory (default **$HOME/<compiler>**)
-t	run some additional tests to check correct installation (a basic encode/decode test is always run)

7. Testing

The GBGP module has a basic user-test option; this is not intended to be a comprehensive test of the software, but just to demonstrate that the package has been correctly installed and programs generate data which can be validated against reference datasets (provided). The tests are run as part of the build process, but can be run manually by:

```
> cd gbgp-<ver>/tests
> make check
```

The screen output (and in some cases log files) should be reviewed to check to whether the test(s) were successful or not. An example log file is provided to show the output for a successful test. (Tip – run the command:

```
> make check 2>&1 | tee check.log
```

to also save a log file which can be examined at leisure.) Individual tests can also be performed by running the test shell script directly:

```
> gbgp_test.sh <test-no>...
```

where <test-no> can be one or more space-separated digits 1–6:

1	test file name against content
2	test BUFR encoder
3	test NOAA CSV-format conversion
4	test UCAR netCDF conversion
5	test dissemination script
6	test database update script

If using the generic **gbuildpack** or wrapper **gbuild_gbgp** scripts, all tests will be performed automatically if the build process completed successfully and before final installation.

Due to last-digit rounding errors between compilers and similar non-significant differences like current timestamps, there might be some small variation between the generated output files and the provided reference files; if 'diffs' are flagged but visual inspection shown nothing significant, the test can be considered as a **PASS**. A **FAIL** may indicate that a run-time look-up file may not have been found; the log output should indicate the cause.

If there were missing dependencies during the build process then the affected tools (Tests 1–4) will not be built and therefore will not be tested. Tests 5 and 6 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.

If shared library versions of dependency libraries have been installed to non-standard locations (which is probable with **netCDF**), then the GBGP application tools – when run stand-alone at the command-line – may fail with messages about missing libraries. In this case, it will be necessary to add the install path(s) to the environment variable **LD_LIBRARY_PATH**; for example:

```
> export LD_LIBRARY_PATH=$GBGP_ROOT/ifort/lib64:$LD_LIBRARY_PATH
```

This environment variable is set appropriately within **gbuildpack** and **gbgp_test.sh** but of course will be lost when the script ends.

8. Installing under Windows with Cygwin

8.1 Introduction

This section is intended to give some tips for using the **Cygwin** system to build the GBGP code on Microsoft Windows platforms. The GBGP package supports Cygwin builds using the GFortran compiler only, though in principle any Windows or Cygwin-compatible F95 compiler can be used if set up to run under Cygwin. See Section 8.6 on other compilers and Section 8.7 for known compatibility issues.

Using Cygwin allows standard Unix/Linux shell (configure) scripts and build tools to be used, and thus simplifies installation support for the Windows platform. Native Windows build support (e.g. batch command or 'nmake' files) is not part of the supplied package. Once Cygwin itself and suitable compilers have been installed, building GBGP is identical to the procedure under Unix/Linux.

The information below is believed to be correct at the time of writing, but systems evolve, and details of installation and usage can and do change from time to time, so if the guidance here appears not to work, check with the online resources.

8.2 Installing Cygwin

Cygwin is a freeware Linux-style environment that runs under Microsoft Windows XP, 2000, Vista, Win7, Win8/8.1 and Win10 (from v1.7, Win95, WinMe and WinNT are no longer supported). Binaries must be explicitly built under Cygwin; native Windows programs cannot be run under it. Similarly, native-built Linux executables will not run under Cygwin.

The simplest interface is via a Command Line (terminal) window, although an X11 GUI interface is also available. For details on downloading, installing and setting up Cygwin via the setup.exe (32-bit) or setup64.exe (64-bit) program see:

`http://www.cygwin.com`

There are useful tips in the FAQ section and a Cygwin User Guide (in PDF) can be downloaded and browsed at leisure.

For 32-bit Windows systems, the Cygwin 32-bit installation via **`setup.exe`** is required. For 64-bit Windows, we recommend installing the 64-bit Cygwin via **`setup64.exe`**, but the 32-bit (setup.exe) is also an option (and necessary if G95 is the desired F95 compiler – see section 8.4.2). In the following, we will assume 64-bit Cygwin and GFortran.

We recommend downloading the Cygwin **`setup64.exe`** to a top-level (but *not* root) directory such as **`C:\cygwin64`**, and run the program from there by double-clicking on its icon. Select the 'Install from Internet' radio button. If you have to go though a firewall, on the next dialogue panel, set the connection to use IE, otherwise 'direct'. Then choose a local mirror site; an ftp one is best for direct connections, though an http one might be needed if going through a firewall and/or using an IE connection.

You then need to select which bits to install. The default is a relatively small core subset. The minimum setup for our purpose is to take the standard defaults in the Base group and also select:

- **`gcc-4`** (Core), **`gcc-4`** (GCC Release 4 compiler), **`gcc4-fortran`**, **`make`**, **`binutils`** from the Devel group;
- **`diffutils`** from the Utils group;

- **`zlib`**, **`hdf5`** and **`sqlite3`** (together with their **`-devel`** packages) from the Libs group.

If you intend to use the very latest GFortran compiler under Cygwin from the project website instead of the Cygwin-installed version (see Section 8.4.1), also select **`GMP`** and **`MPFR`** maths packages from the Libs group. Unless you have a good reason to install the bleeding-edge latest release, we recommend using the official Cygwin version of GFortran.

Some packages may have third-party dependencies which may need other tools; for instance **`netCDF-4`** requires **`HDF5`** which in turn optionally uses **`ZLIB`**. Should you need to re-generate configure or makefiles you will require the **`automake`**/**`autoconf`** toolset from the Devel group.

In order to build against dependency libraries such as **`ZLIB`**, **`SQLite3`**, **`netCDF-4`** and **`HDF5`**, also select their 'development' package in addition to the run-time library in order to install the associated C header (**`.h`**) files.

`Setup64.exe` will then download and install the required and any dependent components. **`Setup64.exe`** can be run again at any time to install other components of Cygwin should they be needed for something else. **`Setup64.exe`** will also detect and install any updated components after the initial install, and it can also be used to uninstall Cygwin components.

Before the installation completes, take the option to place a Cygwin icon on your Desktop; double-clicking on this will open a Cygwin terminal window. This is just a standard Windows Command Line interface window, and it can be configured for colours, font, window size and command history (depending on which Windows version is being used) as for any other 'DOS box'. However, it is running the Cygwin application (Bash shell) and not Windows command shell (aka MSDOS).

The terminal should start in your personal HOME directory and run the **`.bash_profile`** setup script if it exists. In our experience, the resource file **`.bashrc`** (normally the place to define aliases) is not sourced automatically at login, but has to be sourced from **`.bash_profile`** (in which case you might as well edit alias definitions directly into **`.bash_profile`**).

Try the bash commands **`ls`** and **`echo $HOME`** to check correct installation.

8.3 Setting up Cygwin

The Windows environment variable **`%HOME%`** is usually set to your home directory, such as **`C:\Users\<username>`**. Under Cygwin, this Windows name would be mapped by default to **`/cygdrive/c/Users/<username>`** unless explicitly mounted – e.g.:

```
> mount /cygdrive/c /c
```

such that **`$HOME`** will now become **`/c/Users/<username>`**. Once created, this mount point will remain until explicitly unmounted, even across logins and reboots. See the Cygwin User Guide for further details. It is not necessary to mount the logical drive in order to build the package. Cygwin maps all Unix-style paths to Windows equivalents on-the-fly. The **`cygpath`** utility can be used to map between Windows, Unix or mixed directory path syntax styles, but from DLL v1.7 this is not necessary.

If **`HOME`** is not pre-set, then we recommend that it is first set up for Windows. For instance, under WinXP, right-click on 'My Computer', select the 'Advanced' tab and click on 'Environment Variables'; in the upper User section click on 'New' and enter 'HOME' in the top text entry box and a suitable path in the bottom. Typically this might be of the form **`C:\<username>`** (or on Win7 and later, **`C:\Users\<username>`**). Click

OK to save; create the target directory if needed. Any open Cygwin terminal window must be closed and a new Cygwin session started for any Windows environment variable change to take effect. Setting environment variables in Vista or Win7/8/10 is very similar.

This mechanism can be used to create or modify other common (i.e. to Windows as well as to Cygwin) environment variables such as **`PATH`**. Environment settings only used in Cygwin can be set up in the user's personal start-up file **`$HOME/.bash_profile`**

In general, applications compiled under Cygwin can also be run under native Windows in a standard 'DOS box' or a batch command file though file syntax differences may not always be supported.

8.4 Cygwin versions of gcc, Gfortran and G95

The GFortran and G95 Fortran compilers are both available under the GNU free software licence conditions (GPL) as source code or pre-built for a variety of platforms including Windows (stand-alone or under Cygwin). The **`gcc`** C compiler is almost universally available.

8.4.1 GFortran

GFortran is included in Cygwin (and all mainstream Linux distros), though this may not be the very latest release. Where available, we suggest using these distro-specific releases of GFortran for building our application rather than installing the very latest, but the choice is yours. We recommend installing the Cygwin default version (currently v5.3.0 at February 2016) via **`setup64.exe`**, but the very latest release can be obtained from the project website. For the download, see:

http://gcc.gnu.org/wiki/GFortranBinaries

and a guide for installation is at:

http://gcc.gnu.org/wiki/GFortranBinariesCygwin

8.4.2 G95

At the time of writing, G95 development seems to have stalled – or possibly abandoned altogether; the last Cygwin build version being v0.93, July 2010. This is a 32-bit build and although Fortran-only code will compile, link and run normally, object code created with it is incompatible with the 64-bit **`gcc`** builds of third-party (dependency) libraries under 64-bit Cygwin (installed via **`setup64.exe`** or locally built). Therefore if G95 *must* be used, install the 32-bit Cygwin (**`setup.exe)`**.

For the details and downloads, see:

`http://www.g95.org/`

Download the binary Cygwin distribution file (tarball) **`g95-x86-cygwin.tgz`** to the Cygwin root directory (e.g. **`C:\cygwin)`**; in a Cygwin terminal window:

```
> cd /
> tar -zxvf g95-x86-cygwin.tgz
```

this will install into the **`/usr/local`** tree; the **`g95`** main executable will be in **`/usr/local/bin`** which should be in your default **`PATH`**. Then:

```
    > g95 --version
```

will confirm correct installation. Should an updated version become available, simply re-install using the latest tarball.

Note that we do not support the building or testing of GBGP or its dependency libraries with G95 on any platform.

8.4.3 GCC

If you have used Cygwin's **`setup64.exe`** to install **`gcc4-core`** (recommended), then you will have a version of **`gcc`** that is stable and adequate for our package. The installed version can be confirmed with:

```
    > gcc --version
```

If you plan to install the latest GFortran package from the website, you should first install the **`GCC-core`** package with the same version.

8.5 Building packages

When Cygwin and suitable compilers are correctly set up, the GBGP and dependency packages are simply built by starting a Cygwin terminal session (e.g. from the desktop icon). Change directory to wherever the GBGP package file has been downloaded and follow the installation procedure shown on Section 6.

It is recommended that the install target (prefix) be directed to your Cygwin **`HOME`** root space -- i.e. the build will put Fortran module files in **`$HOME/include`** (**`%HOME%\include`**), the package object library in **`$HOME/lib`** (**`%HOME%\lib`**) and example application executable programs in **`$HOME/bin`** (**`%HOME%\bin`**).

Once the application programs have been built under Cygwin, they can be run as for any native 'DOS' command in any Command Line window ('DOS box'), not just at a Cygwin 'terminal' assuming **`$HOME/bin`** (**`%HOME%\bin`**) is in your normal (DOS) **`PATH`** or the explicit path to the executable is given. Note that the G95 Cygwin installation described above by default needs the **`cygwin.dll`** run-time shared library; see the G95 documentation for instructions on how to build without this dependency.

Tip: On Windows XP, environment variables are set permanently via Control Panel> System> Advanced> Environment Variables - for instance set **`PATH`** to **`%HOME%\bin;%PATH%`** in the User section.

8.6 Supported compilers

Only GFortran is supported by the GBGP build system. G95 may work, but it has not been tested and there are known problems building (or testing) dependency packages, so G95 is not recommended. Given that GFortran has no such issues, is 64-bit compatible and can be directly installed by the **`setup64.exe`** installer, we see no reason why that compiler should not be used as a drop-in replacement.

Other compilers may be able to build the package as long as they are correctly set up and can be run in the Cygwin environment and POSIX-style compiler flags are accepted. Note however, that some code in the dependency packages may use compiler directives, so a F95 compiler supporting **`-fpp`** or equivalent pre-processor option is required. Some code also depends on F95 features which may not be supported by older F90-only compilers.

If you have a compiler which works under Cygwin but not known to the package configure script, please notify us of its name, command line invocation, version and any compiler flags necessary for a successful build so that this combination can be added to the script for a future release.

Note that GBGP depends on third-party code (e.g. **`SQLite3`**, **`HDF5`**, **`netCDF-4`**); we cannot guarantee that such external packages support builds under Cygwin with particular compilers other than GFortran noted above. For instance, there are known limitations in the **`configure`** system which make it incompatible with using the Microsoft C++.NET and Intel VF95 or Salford F95 native compilers because either their command-line option syntax is so different from the Unix style or because file naming is incompatible with Unix's standards (see Section 8.7).

8.7 Issues for Cygwin vs. Unix/Linux

Building under Cygwin is little different from building under Unix/Linux, and Unix file naming conventions are employed, not Windows native names, as shown in Table 5.

Item	***Unix/Linux***	***Windows***
Object file	**`<name>.o`**	**`<name>.obj`**
Object library	**`lib<name>.a`**	**`<name>.lib`**
Executables	**`<name>`**	**`<name>.exe`**

Table 5. Differences between Unix/Linux and Windows file names

The supported compiler GFortran and the GNU build utilities use the Unix-standard **`.o`** and **`lib<name>.a`** convention even on Windows. Depending on the package. the **`.exe`** extension may or may not be added to executable programs.

Native Windows compilers use **`/option=<opt>`** syntax while **`configure`** expects **`-option <opt>`**, though the Windows Intel compilers accept both styles in most cases. Worse, the Salford F95 compiler by default only compiles source code and needs a switch to generate executable code; this is contrary to the traditional use **`-c`** (or **`/c`** for VF95) to 'compile-only'.

These limitations make using the Salford F95, Intel VF95 & MS C++ compilers – and possibly other Windows native compilers – unsupported for our package as the **`configure`** scripts will fail to detect what is otherwise a perfectly good working compiler.

All things considered, we recommend using the 64-bit Cygwin-package version of GFortran. It only needs a simple installation via **`setup64.exe`**, is a mature project with continuing development and support and the package (with all dependencies) has been successfully tested with this compiler on Cygwin (and Linux).

Finally, it is our opinion that Windows/Cygwin is anyway not a suitable platform for production/operations, but ought to be limited to development and testing and portability checking only.

9. Portability

A prime requirement of GBGP is that it must be portable so as to compile and run under a range of compilers and platforms. Nevertheless, GBGP can only be supported on platforms also supported by the required third-party packages. In practice this means POSIX-compliant systems (Unix, AIX, Linux, MS Windows/Cygwin, SunOS, OS/X, HP-UX...) and ISO-compliant F95 & C compilers. GBGP depends on some F95 features, so older F90-only compilers cannot be supported. In addition, the GBGP code can only be fully tested and validated on platforms and with compilers available to the GBGP Development Team.

Target compiler/platform combinations we aim to support are listed in the following tables (with varying dgrees of success). Difficulties have been encountered with certain compilers, mostly due to explicit or implicit bugs in the compilers themselves, or some limitations in their functionality or interpretation of Fortran standards. The investigation of workarounds continues, although it is not our policy to provide different code for different compilers just because certain compilers do not adhere to accepted standards or have internal bugs.

For this release, the OS/compiler combinations shown in Table 6 have passed all of our comprehensive internal testing ('Test Folder' system) and so should work on similar systems elsewhere. Unless noted, this release of GBGP works with **`zlib`** v1.2.8, **`hdf5`** v1.8.16, **`netcdf-c`** v4.4.0, **`netcdf-fortran`** v4.4.3, MetDB **`bufr`** v24.0.2, and **`sqlite3`** v3.13.0**.** The RHEL6 platform is a 64-bit OS with 64-bit compilers (unless noted). The GNU **gcc** compiler is used for compiling C code except where noted.

Platform	***Compiler Vendor***	***Compiler ID***	***Command***	***Version***	***Notes***
Linux (RHEL6.8) Intel Xeon	Intel	ifort13	**`ifort`**	v13.1	GCC v4.4.7
Linux (RHEL6.8) Intel Xeon	Intel	ifort16	**`ifort`**	V16.0	GCC v4.4.7
Linux (RHEL6.8) Intel Xeon	Portland Group	pgf15	**`pgf95`**	v15.7	GCC v4.4.7 See Note 1
Linux (RHEL6.8) Intel Xeon	NAGWare	nagfor60	**`nagfor`**	v6.0	GCC v4.4.7
Linux (RHEL6.8) Intel Xeon	GNU GFortran	gfortran	**`gfortran`**	V4.4.7	GCC v4.4.7
Linux (RHEL6.8) Intel Xeon	Oracle (SUN)	sunf95	**`sunf95`**	v8.6	Oracle SolarisStudio 12.3 32-bit compilers. **suncc** v5.12 See Note 2
Linux (Cray LE) HPC	Cray	ifort15	**`ifort`**	v15.1	GCC v4.9.1 See Note 3
Linux (Cray LE) HPC	Cray	gfortran	**`gfortran`**	v4.9.1	GCC v4.9.1 See Note 3
Linux (Cray LE) HPC	Cray	ftn	**`ftn`**	V8.3.4	GCC v4.9.1 See Note 3

Table 6. OS/compiler combinations passing the Test Folder system

For this release, the OS/compiler combinations shown in Table 7 have failed some element of our comprehensive internal testing of GBGP library code and/or application tools ('Test Folder' system). The OS/compiler combinations shown in Table 8 have failed on one or more external dependency packages. Naturally, if an external dependency cannot be built, neither can any GBGP tool based on it, unless system-built (distro) version is (or can be) installed.

Platform	***Compiler Vendor***	***Compiler ID***	***Command***	***Version***	***Notes***
					No known issues

Table 7. OS/compiler combinations which have failed in some part of the GBGP Test Folder system

Platform	***Compiler Vendor***	***Compiler ID***	***Command***	***Version***	***Dependency Package***	***Notes***
Linux (RHEL6.8) Intel Xeon	Portland	pgf11 pgf15	**`pgf95`**	V11.1 v15.1	NetCDF-4 Fortran API v4.4.3	NetCDF-Fortran test program **`nf_test`** fails to compile with unresolved references to **`exit_()`** See Note 1

Table 8. OS/compiler combinations which fail to build an external dependency package

In addition, all dependency code and GBGP modules have been successfully built and passed basic user-tests (as provided in the distribution tarball) under the following OS/compiler combinations shown in Table 9, unless explicitly noted otherwise.

Platform	***Compiler Vendor***	***Compiler ID***	***Command***	***Version***	***Notes***
Linux (RHEL6.8) Intel Xeon	Intel	ifort12	**`ifort`**	v12.0	GCC v4.4.7
Linux (RHEL6.8) Intel Xeon	Intel	ifort14	**`ifort`**	v14.0	GCC v4.4.7
Linux (RHEL6.8) Intel Xeon	Intel	ifort15	**`ifort`**	V15.1	GCC v4.4.7
Linux (RHEL6.8) Intel Xeon	Portland Group	pgf11	**`pgf95`**	v11.7	GCC v4.4.7 See Note 1
Linux (RHEL6.8) Intel Xeon	NAGWare	nagfor52	**`nagfor`**	v5.2	GCC v4.4.7
Linux (RHEL6.8) Intel Xeon	NAGWare	nagfor53	**`nagfor`**	v5.3	GCC v4.4.7
Linux (OpenSUSE Leap 42.1) Intel i7-4910MQ	Intel	ifort	**`ifort`**	v14.0	GCC v4.8.3
Linux (OpenSUSE Leap 42.1) Intel i7-4910MQ	GNU GFortran	gfortran	**`gfortran`**	v4.8.3	GCC v4.8.3
Linux (OpenSUSE Leap 42.1) Intel i7-4910MQ	Oracle (SUN)	sunf95	**`sunf95`**	v8.7	Oracle SolarisStudio 12.4 32-bit compilers. **`suncc`** v5.13 See note 2
Windows 10 Pro (Cygwin 64-bit) Intel i7-4790K	GNU GFortran	gfortran	**`gfortran`**	v5.3.0	GCC v5.3.0

Table 9. Additional OS/compiler combinations which have passed the basic user-installation tests

Notes:

1. One or more test programs in the netCDF-Fortran interface package fail to run to completion using any of our Portland Group compilers, but it can be installed with a manual **`make install`**, allowing the GBGP pre-converter tools to be linked with it which then correctly convert netCDF input files to COST-format files.
2. The **`suncc`** compiler does not support shared libraries (according to the **`configure`** tests); dependency libraries are therefore built in static versions only.
3. The Cray **`cc`** compiler fails in the **`configure`** step of all dependency packages using this method; the **`autotools`** macros include a **`-stdc99`** command line option which **`cc`** rejects as an unknown option. The GNU **`gcc`** can be substituted without any problems.

10. Known problems

The following issues have arisen during testing and review, but are not considered blockers to this release and if possible will be addressed for a future release. Note that problems building dependency packages with particular compilers are outside the responsibility of the ROM SAF.

- Cygwin **`g95`** is known to fail when building test programs for several dependency libraries. As development of this compiler seems to have ceased, we do not support this combination. For preference, use **`gfortran`** on Cygwin (part of the GCC package installed with **`setup64.exe`**), which builds and runs all tests normally.
- Similar remarks apply to the other failures in building the external dependency packages as detailed in Section 9.

11. Limitations

- The principle objective of GBGP is to support generation of BUFR files for near-real time dissemination of processed GB-GNSS (ZTD) data to NWP users via the GTS. While the primary application tool provides the means to encode BUFR from the E-GVAP COST-format files, a limited number of other formats are supported by pre-converter tools. There are currently no plans to extend the package to cover downstream scientific use of such data (unlike ROPP's FM and 1DVAR modules).
- Only minor updates to the package are foreseen. Support for previous versions will be limited to problems which can also be reproduced with the latest version. We recommend that users should upgrade to the latest version when released. In general, only the current (*viz* GBGP-1.x) release will fully supported.
- All GBGP stand-alone tools can optionally use a GNSS station metadata SQLite database as part of the quality control step of the input data. The principle purpose of using this database is to detect, flag and reject cases of duplicated station 4-character identifiers. This is not fool-proof and may result in a conservative false-positive. In such cases, the user must independently determine whether the anomalous parameters are incorrect in the input file or in the database. This checking can be disabled (see the User Guide), but then the onus is on the data provider to ensure that the station IDs are unique to the actual site they are processing.

12. And finally...

Source files, data files, documentation, etc., in this *prototype* release may be updated from time to time. The software is not guaranteed to be bug-free, but has been tested with operational E-GVAP, NOAA and UCAR data and is running under a quasi-operational environment at the Met Office. The code is fully-functional and has passed a limited set of tests but the complete package has not yet been comprehensively tested or formally reviewed and approved for general release; as such this release has *prototype* status.

We encourage you to carefully check the building and installation, to test the running of the GBGP software and to review the package in its entirety. The GBGP Development team welcomes any feedback if there were any problems, or just to confirm there were no issues.

When the package has been formally approved for full *operational* release, updates to the GBGP code to:

- enhance functionality (for instance to add support for new input sources);
- extend the range of supported WMO GTS routing headers;
- fix any reported bugs;
- extend portability;
- improve the user-build experience;
- correct or clarify any reported issues in the documentation

will be released from time to time as minor releases. Registered users will be notified when an update is available. This may be in the form of a complete replacement file, a patch file or – in the case of trivial changes – instructions for local file editing. Implementation of such updates is at the discretion of the user, but support for older releases will be very limited.

All comments on, or questions about, the GBGP software should, in the first instance, be reported via the ROM SAF Helpdesk at http://www.romsaf.org **`> Helpdesk > New enquiry`**