Open Navigation Surface Working Group

Meeting

Summary

2022-06-09

Canadian Hydrographic Conference

Gatineau, Canada

DRAFT VERSION

2022-07-12

# Introduction

This document details the result of the meeting held on 2022-06-09 to continue development and maintenance of the library. The meeting was held at the Canadian Hydrographic Conference in Gatineau, Canada. The summary of all meetings and teleconferences of the Open Navigation Surface Working Group (ONSWG) can be obtained from the project’s web-site, http://www.opennavsurf.org. For a list of participants, see section 4.

In the following, names of people with action items are shown in **Bold Small Caps**; expected deadline release dates are shown in red. Sizes of variables are indicated by ‘U’ for unsigned, ‘S’ for signed, ‘F’ for floating-point, and a size in bits (e.g., U8 is an eight-bit unsigned integer, F64 is a 64-bit (double precision) floating-point number). Data sizes are given in bytes (B) with the usual convention that the SI multipliers are taken to mean multiples of 210B (i.e., 1kB = 210B = 1024B). The acronym ‘CR’ means ‘Candidate Release’ (i.e., a release of the library for comments) and ‘FR’ means ‘Full Release’.

# Summary of Discussion

## Vertical CRS Specification

The group discussed the method for specification of vertical coordinate reference systems, and coordinate reference systems in general, in the BAG file. This is something currently done through the XML metadata element, and is only loosely specified. Poorly controlled WKT strings are currently used. The request from multiple parties was that we regularize and codify exactly how the CRS was to be specified, so as to avoid significant variation (and therefore interpretation) between parties. A working document had been circulated prior to the meeting to identify some of the issues.

A lengthy, technical, and lively, discussion followed the Chair’s introduction and opening comments from **Rice** [NOAA] outlining the requirement for the current request for clarification due to the on-going National Bathymetric Source project at NOAA Office of Coast Survey. A fundamental question [**Paton**] is whether it is sufficient to simply specify the datum of the data encoded in the BAG file, or if it was required to record the transformation(s) of datum that were used to achieve the data (and therefore how it could be converted to a pivot datum [**Vautour**] for further manipulation). An associated difficulty is that there are often not good definitions of the large variety of vertical datums used in practice [**Rice**, **Renoud**], or an authority that registers them so that a reference number could be used, rather than a full definition encoded in the BAG file. It might also be difficult, given the level of bureaucracy involved, to get new definitions registered somewhere, making the process slow. It was also felt [**Calder**] that providing the facility to fully specify the datum transformation information within the BAG file could potentially increase the complexity of the BAG file and library significantly, contrary to the general principle of the BAG specification. This would be particularly true if the WG had to become a *de facto* registry of datums and transformations. It would also be the case [**Riley**, **Paton**] that comprehensibility and discovery might be impacted by a more complex encoding scheme.

The group discussed the scope of the problem, and in particular whether it was sufficient to provide just the lineage of transformations used, or if a “working” definition (i.e., how one might transform from the given data to something else) was required [**Vautour**]. Given the complexity, it was concluded that it would probably be better – for now – to just define the datum for the current data, and provide the option to include transformation data, in a separate item, if required. Then, people who “just want to use the data” can read the core information, and those with more technical requirements can delve deeper, if the optional data is provided. A more complex and comprehensive solution might be considered in the future, but not for the current round of updates to the project.

A number of different approaches were suggested for how the datum might be defined, including EPSG codes; WKT2 or WKT1 text; a Bound CRS (which includes a datum and transformation); and a gridded separation surface layer (possibly re-purposed from the current optional separation layer). It was felt [**Paton**, **Vautour**] that the EPSG code, since it could be looked up canonically, would generally be preferred, but [**Vautour**] it would be necessary to allow for WKT as a backup, since not all datums or (optional) transformations have known EPSG codes. It was also suggested [**Paton**] that a single separation surface might be the simplest solution from a code development point of view, since it always contained a single step transition from a known datum to the current datum. Although this would require an extra data burden, it was pointed out [**Calder**] that the separation grid is typically low resolution and only for a limited area, and therefore the data burden might be quite small in comparison to the rest of the data in the file. It was suggested [**Rice**, **Masetti**] that whatever method is used, it would be very important to ensure that validation checks are carried out to demonstrate that the metadata is correct and complete.

A question of backwards compatibility [**Masetti**] was posed: should we maintain use of the current metadata elements, or change to new elements to make explicit that the usage has changed? There is clearly a concern [**Johnson**] for the costs and effort involved in something that would require archives of BAGs to be updated. In discussion, it became clear that the current situation is sufficiently uncontrolled that the changes being proposed would still fit within current usage, and therefore should not pose a backwards compatibility problem. It was agreed, however, that it would probably be necessary to indicate in the metadata that the new methods were being used so that library implementations that were aware of the new strictures could tell that the metadata should follow the new norms. It as suggested [**Calder**] that a minor version step could be used for this.

As a synthesis of the discussion, a proposed solution [**Calder**] was outlined, and then modified by the members of the group. The final proposition was:

1. Define in the FSD a formalization of the XML fields that are used to specify the horizontal and vertical CRS in the BAG file. The goal being to make sure that there is no ambiguity in how these fields should be filled out in the future.
2. Provide a definitive and axiomatic formalization of the encoding in the CRS fields (both vertical and horizontal).
3. If a Geographic 3D CRS is specified, mandate that it only appear in what is currently the “horizontal” CRS position, and that the “vertical” CRS metadata must be blank. Failure to do this is a validation error.
4. Strongly prefer EPSG codes when possible, and prefer the EPSG code embedded in WKT over the WKT itself. Use WKT as a backup if the EPSG doesn’t exist.
5. Allow optional transformation information to be encoded, but in a separate metadata element from the horizontal/vertical CRS information. Provide a definitive and axiomatic encoding of same.
6. Provide for validation of the CRS and (optional) transformation information through a suitable mechanism (e.g., XML schema).

It was not clear, however, from the extensive discussion, exactly how to execute this general proposition, and the group therefore concluded that a sub-working group should be assembled to investigate the issues raised by the discussion, and to report to the overall working group in a short period. The Chair agreed to provide a charge for the group [Action: **Calder**], and the group co-opted **Rice** and **Riley** (NOAA), **Renoud** (QPS), **Vautour** (Teledyne CARIS), and **McCooey** (Leidos) for the sub-WG, with **Rice** to chair [Action: **Rice**].

## Integration of the v2.0 API

The group discussed the creation of a pull request for the developed, but never merged, V2.0 API for BAG files. This branch of the repository was developed pre-pandemic, but never fully tested or merged. It is a significant advance on the current API, however, and supports the metadata layers required for NOAA’s National Bathymetric Source project. CCOM volunteered [**Calder**] to provide technical support and staff time to make the branch into a pull request, and to merge, and requested consensus from the group to that effect. There was confusion over whether this would change the binary structure of the BAG file, and the effects if so on backwards compatibility. It was agreed, however, that the binary structure would not be changed, except with the addition of the optional metadata layer, which readers simply ignore if they do not understand it. It as pointed out [Calder] that this does mean that we have to track separately the binary version for the BAG file, and an API version for the library [Action: **Calder**]. With this understanding, the group consented to moving ahead with the merge [Action: **Calder, Rice**]. There was a request [**Masetti**] to attempt to separate the API update and NOAA-specific metadata layer addition into two pull requests if possible [Action: **Calder**].

Further discussion of maintenance of the current API, and a timeline for transition then occurred. It was pointed out [**Byrne**] that there is a non-trivial cost to re-code applications to use the new API, and therefore that some leniency in transition time might be required. The group agreed that there was no way to force a particular vendor to switch to the new API, and since the binary format of the BAG file would not be changed by these actions, it would be entirely feasible for vendors to stick with the older API for as long as required. It was pointed out [**Calder**], however, that there is a non-zero cost in maintaining two versions of the library, and that therefore that transition to the new API should be considered mandatory for new applications, and strongly encouraged for upgrades to old ones. Consideration was given to updating the support libraries used for the BAG file, which are now somewhat aged [**Paton**], but it was agreed that this would certainly make a significant difference to backwards compatibility, and therefore that it was outside the scope of the current proposal (although it should remain on the agenda for future development).

The group then discussed requirements for the process, specifically the OS, support library, and compiler versions that could be used. It was agreed that support for Windows 10, Red Hat Enterprise Linux 8, and macOS 11.6 were required; C++17 was acceptable as a goal; CMake 3.11 or higher for the build system; and although compilers vary with OS, at least MinGW/Win10, MSVC/Win10, gcc/Linux, and clang/macOS would be required. It was pointed out [**Johnson**] that some developers have heavier constraints on which library and compiler versions are allowed or supported due to security issues; these constraints are at present unclear, but will be confirmed [Action: **Johnson**].

## Integration of the Crypto++ Library for Digital Signatures

The initial specification for the BAG file included a digital signature scheme to demonstrate authorization and integrity of the data. Although this service has never been widely used, the group agreed at the previous meeting to continue to support the facility, particularly as data security is becoming more of a concern in the hydrographic world. The library used to provide the core services (BeeCrypt) is now, however, significantly out of data, and no longer in active maintenance.

The group therefore discussed swapping to the Crypto++ library, which provides equivalent services and is more frequently updated. A pull request for this has already been submitted, but never merged. After some discussion, the group agreed that this could move ahead, although with the caveat that compilation of the library remains optional as at present [Action: **Calder**].

## General Library Building Processes

The group briefly discussed the general build process for the library. Currently, the library is distributed with dependency libraries, mainly to ensure that specific versions are used for compilation; it was questioned [**Masetti**] whether we should continue to do so. It was suggested [**Paton**] that it might be possible to have CMake pull down specific versions of dependent libraries, which would simplify things; it was however noted [**Johnson**] that this might be difficult is more secure environments. It was agreed that this would be investigated as part of the overall pull request process outlined in this meeting [Action: **Calder**].

There was then a brief discussion about the potential for continuous integration and binary distribution [**Masetti**]. It was noted [**Calder**] that CI is now being used within CCOM where the pull requests being discussed will be prepared, and that if possible, the binaries so produced would be made available for distribution, albeit with no guarantees as to compatibility with any particular system [Action: **Calder**].

## AOCB

The Chair thanked all of the developers present, and their organizations, for finding the time to conduct the meeting, and the Conference organizers for providing the meeting space.

# Summary of Action Items and Dates

The following actions and dates were agreed:

| Person | Actions(s) | Section | Date |
| --- | --- | --- | --- |
| Calder | Provide charge for sub-working group on CRS definition | 2.1 | 2022-07-01 |
| Rice | Chair sub-working group on CRS definition | 2.1 | 2022-08-12 |
| Calder | Ensure that BAG file and library versions are defined separately and documented in the FSD. | 2.2 | 2022-09-01 |
| Calder, Rice | Generate pull request for v2.0 API, orchestrate review, and merge | 2.2 | 2022-09-01 |
| Calder | Attempt to make v2.0 API into two pull requests (one for API, one for metadata layers) if possible | 2.2 | 2022-09-01 |
| Johnson | Provide list of latest versions of libraries that are supportable within the NAVOCEANO environment. | 2.2 | 2022-08-12 |
| Calder | Orchestrate review of Crypto++ pull request, and merge, maintaining optional compilation of the library. | 2.3 | 2022-09-01 |
| Calder | Consider whether CMake could, optionally to support more constrained networks, be used to pull depenent libraries and compile them. | 2.4 | 2022-09-01 |
| Calder | Make binary distribution artefacts from CI builds available for download if possible. | 2.4 | 2022-09-01 |

# Participants

Jonathan Beaudoin (QPS)

Steve Brooker (QPS)

Shannon Byrne (Leidos) [Zoom]

Brian Calder (CCOM/JHC)

Burns Foster (Teledyne CARIS)

Tami Franksen (Teledyne CARIS)

Grant Froelich (NOAA)

Barry Gallagher (NOAA)

Stacey Johnson (NAVOCEANO)

Giuseppe Masetti (CCOM/JHC; Danish Hydrographic Service)

Chris McCooey (Leidos)

Kelly Nifong (NAVOCEANO)

Mark Paton (QPS)

Weston Renaud (QPS)

Glen Rice (NOAA)

Jack Riley (NOAA)

Matt Thompson (NAVOCEANO)

André Vautour (Teledyne CARIS)

Matt Wethers (NAVOCEANO)

Eric Younkin (NOAA)

Chen Zhang (NOAA)