

Work Item H

Data Life Cycle

Report to CSBWG16

NIWA, Wellington, NZ

27 March 2025

Brian Calder

CCOM/JHC, University of New Hampshire



IHO

International
Hydrographic
Organization



IHO

Work Item Team Members

International
Hydrographic
Organization

- Brian Calder (CCOM/JHC)
- Clint Campbell (NCEI)
- Knut Hartmann (EOMAP)
- Brian Jensen (Raymarine)
- Giuseppe Masetti (GST)
- Brian Miles (CCOM/JHC)
- Guillaume Morissette (CIDCO)
- Mathieu Rondeau (CHS)
- Thierry Schmitt (SHOM)
- Shaul Solomon
- Colin Thomson (OFM)
- Emma Wise (Teledyne Marine)



IHO

Work Item Background & Purpose

International
Hydrographic
Organization

This work item is intended to provide input on [details of data treatment, processing, and management], essentially **providing clarification** to the high-level description of the process in B-12. The work item is **not intended to modify or update B-12**, but rather to provide input to this process, **making recommendations** for required updates, and providing specific solutions for problems encountered in practice where possible.

**IHO**

Progress Since CSBWG15

International
Hydrographic
Organization

Work Item	Title	Priority	Next milestone	Start Date	End Date	Status
1	Recommendations for DCDB data access	H	CSBWG15 Intersessional	2024-06-01	2024-08-28	C
2	Consensus on workflow for developers	M	CSBWG16	2024-04-26	N/A	O
3	Consensus on workflow for end-users	M	CSBWG16	2024-04-26	2025-06-30	O
4	OpenVBI code review and development	H	CSBWG16	2024-04-26	2025-03-30	O
5	Vertical bias corrections in OpenVBI	L	CSBWG16	2024-04-26	2025-06-30	O
6	Uncertainty estimation in OpenVBI	L	CSBWG16	2024-04-26	N/A	O
7	Merge activities of sub-group (work item) F.	M	CSBWG16	2024-08-01	N/A	O



IHO

2&3: Consensus on Workflow

International
Hydrograph
Organization

Rondeau et al., From Volunteer Ping to Community Map: The CHS'
Community Hydrography Program, IHR 30(1), 2024
(https://doi.org/10.58440/ihr-30-1-a07)

International Hydrographic Organization

Past issues
Volume 30(1)
Volume 29(2)
Volume 29(1)
All issues →

Collections
Bathymetry and
Positioning
Research and Technology
All collections →

About
Aims & Scope
Editorial Board
Publication Policies &
Deadlines
Peer Review Process
Visit IHO website

Submissions
Submit an Article
Author Guidelines
Reviewer Guidelines

Fig. 12 Map of tide gauge stations and their respective areas of influence (Voronoi polygons) for all of Canada. Green + stations for which predictions are available. Pink + stations for which observations are available.

Fig. 13 Bounding polygon of the dataset to be reduced, superimposed on the global influence map. Highlighted are the Voronoi polygons solicited by the dataset to be reduced.

International Hydrographic Review
Volume 30(1)

Currently reading
From Volunteer Ping to Community Map - The CHS' Community Hydrography Program

Cite Share Download

Previous article Next article

anthonyklemm / csb_processing_pipeline

Code Issues Pull requests Discussions Actions Projects Security Insights

csb_processing_pipeline Public

Watch 1 Fork 0 Star 1

main 1 Branch 0 Tags

Go to file Add file Code

About
No description, website, or topics provided.
Readme
CC0-1.0 license
Activity
1 star
1 watching
0 forks
Report repository

Releases
No releases published

Packages
No packages published

Languages
Python 100.0%

1-csb_scraper.py	Update and rename 1-csb_scraper_mod1.py to 1-csb_scr...	4 days ago
10-csb_differencing_visualizations.py	Update 10-csb_differencing_visualizations.py	4 days ago
2-csb_processing.py	Update 2-csb_processing.py	3 days ago
3-load_csb_to_duckdb.py	Initial commit	last week
4-histograms_and_calibration_points.py	Initial commit	last week
5-apply_best_offsets_duckdb.py	Update and rename 5-apply_best_offsets_duckdb 2.py to...	4 days ago
6-export_transits_to_gpkg_and_tiff_2.py	Initial commit	last week
6-export_transits_to_gpkg_and_tiff_speed.py	Update 6-export_transits_to_gpkg_and_tiff_speed.py	4 days ago
7-Outlier_model_PMM_imputation.py	Update and rename 7-Outlier_model_PMM_imputation_it...	4 days ago
8-insert_outlier_flags_in_duckdb.py	Update and rename 10-insert_outlier_flags_in_duckdb.py...	4 days ago
9-csb_export_all_points_create_geotiff.py	Update and rename 11-csb_export_all_points_create_geo...	4 days ago
LICENSE	Initial commit	last week
README.md	Update README.md	3 days ago
build_leaderboard_and_tracklines	Update and rename 7-build_leaderboard_and_tracklines ...	4 days ago
count_number_outliers_duckdb.py	Initial commit	last week
csb_differencing_visualizations_work.py	Initial commit	last week
csb_export_all_points.py	Initial commit	last week
dashboard 1.py	Update and rename 8-dashboard 1.py to dashboard 1.py	4 days ago

README CC0-1.0 license

CSB data pipeline for scraping, tide correction, spatial database population (duckdb-spatial), vertical offset analysis/correction, uncertainty estimation, vessel speed estimation and PMM imputation-based outlier detection algorithms, and exporting data as vessel-transit geopackages and geotiff DEMs.

There are other helper scripts, and some scripts for dashboard/leaderboard creation as well.

You're going to need a shapefile of the CO-OPS discrete zoned tide model. One is provided in the OCS Pydro



IHO

4: OpenVBI Code Review and Development

The screenshot shows the GitHub repository for OpenVBI, maintained by CCOMJHC. The repository is public and has 7 forks and 0 tags. The main branch is selected. The file list includes .devcontainer, openvbi, .gitignore, Dockerfile.build, LICENSE, README.md, pyproject.toml, requirements-build.txt, requirements.txt, setup.cfg, and setup.py. The README.md file is open, showing the project's purpose: "Reference Algorithms for Volunteer Bathymetric Information processing." It includes an "Installation" section with "Local installation" (using pip install) and "Running in a VS Code dev container" (using Docker). The right sidebar shows the repository's activity, including commits, releases, packages, contributors, and languages.

The screenshot shows a GitHub Discussion thread titled "CSB/VBI Tools Workshop at CSBWG16 #2" initiated by user brian-r-calder. The discussion is categorized as "Ideas" and has 6 participants. The thread content includes:

In addition to the IHO [working group meeting](#) in Wellington, NZ (26-28 March 2025), there is going to be a [practical workshop on CSB/VBI tools \(24-25 March 2025\)](#) in the same venue. The first day of the workshop is going to have "state of the art" assessments on the various stages of data capture and processing by various experts, along with an expert panel and hands-on time with hardware and software data collection and processing systems. The second day of the workshop will have on-the-water demonstration time, but also a full-day development session (hackathon) for developers to work with different open source toolsets for dealing with CSB/VBI data. Details in the attached flier.

The development workshop is being supported by CCOMJHC at the [University of New Hampshire](#), and we're looking for input on a number of different parts of the second day's events. Particularly:

1. We want to do some actual development work, so we'll need a base library on which to work. What library do we want to use as a base? We're partial to OpenVBI (of course), and it would make a good development base, but are there other libraries that we should consider?
2. We want to build and test code. What development environment do we want? We're thinking Python 3.12, Git, and GitHub (and maybe VSCode). But should we maybe consider working in a development container, or some other structure?
3. We're likely to have some people participating remotely so we'll need some sort of collaboration tool. There are any number of collaboration platforms from Teams to Discord to Slack. Is there a favourite, or a preferred solution that would provide GitHub integration, messaging, and video conferencing?

There are probably many other things to discuss, but let's start the conversation here, and see how it goes. There's also a [very simple website](#) to provide some more details on the workshop, and links for registration, background documents, etc.

[CSB_Event_NewZealand_2025.pdf](#)

The right sidebar shows the discussion's metadata, including the category "Ideas", labels, participants, and notification options.



International
Hydrographic
Organization

IHO

5: Vertical Bias Corrections in OpenVBI

```
openvbi > examples > prep-simple.py > ...
1 from openvbi.adapters.ydvr import load_data
2 from openvbi.filters.thresholding import shoaler_than, deeper_than
3 from openvbi.filters.timeslot import before_time, after_time
4 from openvbi.corrections.waterlevel.noaa import ZoneTides
5 from openvbi.adapters.dcdb import write_geojson
6
7 # Pull in data from YachtDevices raw binary file, and convert to depths assuming NMEA2000
8 data = load_data('00030095.DAT')
9 data.generate_observations('Depth')
10
11 # Calibrate acceptable depth and time windows for data (note that these are simply for
12 # demonstration purposes: this cuts off a lot of valid data!)
13 min_depth = data.depths['z'].min()
14 max_depth = data.depths['z'].max()
15 depth_range = max_depth - min_depth
16 shoal_threshold = min_depth + depth_range/3.0
17 deep_threshold = max_depth - depth_range/3.0
18
19 min_time = data.depths['t'].min() + 10.0*60.0 # Remove first ten minutes
20 max_time = data.depths['t'].max() - 10.0*60.0 # Remove last ten minutes
21
22 # Generate filters for shoal/deep depth, and early/late time
23 shoal = shoaler_than(shoal_threshold)
24 deep = deeper_than(deep_threshold)
25 early = before_time(min_time)
26 late = after_time(max_time)
27
28 # Filter for depth window, and time window
29 data = early.Execute(late.Execute(deep.Execute(shoal.Execute(data))))
30
31 # Correct for waterlevel using NOAA zoned tides and Live API for waterlevels
32 zone_tide_wl = ZoneTides('NOAA_tide_zones/tide_zone_polygons_new_WGS84_merge.shp')
33 zone_tide_wl.preload(data)
34 data = zone_tide_wl.correct(data)
35
36 # Generate B.12-format GeoJSON output
37 write_geojson(data, '00030095.json', indent=2)
38
```

```
openvbi > corrections > waterlevel > noaa > __init__.py > ...
108 class ZoneTides(Waterlevel):
109     def __init__(self, zone_shapefile: str) -> None:
110         self._zones = geopandas.read_file(zone_shapefile)
111         super().__init__()
112
113     def preload(self, dataset: Dataset) -> None:
114         # Spatial join to determine which polygon each observation is in (and hence which
115         annotated_pts = geopandas.sjoin(dataset.depths, self._zones, how='inner', predicate='contains')
116         # List of all required stations
117         self._stations = annotated_pts['ControlStn'].unique()
118         self._tides = dict()
119         # For each station, we need to determine the time bounds of the observations affect
120         # call the CO-OPS API to get the waterlevel corrections; these are stored until it
121         # do the corrections for some/all of the observations
122         for station in self._stations:
123             station_times = annotated_pts[annotated_pts['ControlStn'] == station]['t']
124             min_time = station_times.min() - 10*60
125             max_time = station_times.max() + 10*60
126             raw_levels = get_noaa_station(station, min_time, max_time)
127             corrections = InterpTable(['dz',])
128             for n in range(len(raw_levels)):
129                 corrections.add_point(raw_levels['t'][n].timestamp(), 'dz', raw_levels['v']
130                 self._tides[station] = {'min': min_time, 'max': max_time, 'raw': raw_levels, 'corr': corrections}
131
132     def _execute(self, observations: geopandas.GeoDataFrame) -> geopandas.GeoDataFrame:
133         # Spatial join to determine which polygon each observation is in (and hence which
134         annotated_pts = geopandas.sjoin(observations, self._zones, how='inner', predicate='contains')
135         for station, data in self._tides.items():
136             station_points = annotated_pts[annotated_pts['ControlStn'] == station]
137             lut_times = station_points['t'] - station_points['ATCorr']*60
138             wl_corr = data['table'].interpolate(['dz',], lut_times)[0]
139             station_points['z'] = station_points['RR']*wl_corr
140             observations.loc[station_points.index, 'z'] = station_points['z']
141         return observations
142
143     def _metadata(self, meta: md.Metadata) -> None:
144         meta.addProcessingAction(md.ProcessingType.VERTREDUCTION, None,
145         reference='ChartDatum',
146         datum='MLLW',
147         method='Observed Waterlevel',
148         algorithm='OpenVBI',
149         version=version(),
150         model = f'NOAA Zoned Tides with stations {self._stations}')
```



IHO

6: Uncertainty Estimates in OpenVBI

International
Hydrographic
Organization

- CS
- The
- Res
- Co
- Imp
- Inte

```
examples > datalogger_examples.py > main
5 def report_uncertainty(label: str, tpu: np.ndarray[float]) -> None:
15     print(f"Approximate y error: {error_y:.3f} m")
16     print(f"Approximate z error: {error_z:.3f} m")
17
18     print("\n-----\n")
19
20 def uncrnt_m_to_radians(u: float) -> float:
21     return u * 1.0 / (1852.0 * 60.0) * (math.pi / 180.0)
22
23 def main() -> None:
24     # Common simulated data used for all scenarios here
25     observation: Measurement = Measurement(latitude=math.radians(48.4525), longitude=math.radians(-68.5232), height=math.radians(350), pitch=math.radians(0.5), roll=math.radians(1.5), depth=5.0)
26
27     offset: Offset = Offset(x=1.0, y=2.0, z=3.0)
28
29
30 # The TPU class is just functionality (i.e., has no internal state), so we can generate one
31 # for all of the computations that we're going to do.
32 u = TotalPropagatedUncertainty()
33
34 # -----
35 # High-accuracy CIDCO datalogger
36
37 #Centimetric accuracy is typical when using PPP
38 #IMU accuracy: the BN0055 datasheet gives +-1 degree on roll/pitch, and +-2 degrees on heading. We'll assume t
39 hydroblock_meas_u: Measurement = Measurement(latitude=uncrnt_m_to_radians(0.02), longitude=uncrnt_m_to_radians(0.02), height=math.radians(2.0/2.0), pitch=math.radians(1.0/2.0), roll=math.radians(2.0/2.0), depth=0.05)
40
41 #Offsets are measured with a total station, with millimetric accuracy
42 hydroblock_off_u: Offset = Offset(x=0.001, y=0.001, z=0.001)
43
44 tpu = u.compute(observation, offset, hydroblock_meas_u, hydroblock_off_u)
45 report_uncertainty("Hydroblock Uncertainty", tpu)
46
47
48 # -----
49 # Generic data logger with boat's GNSS/Depth (e.g., TeamSurv, YDVR04, or WIBL)
50
51 # GNSS horizontal order 1m is feasible-most places with SA off and a reasonable receiver
52 # Heading/Pitch/Roll are typically not measured, so uncertainty could be up to the expected motion
53 vbi_meas_u: Measurement = Measurement(latitude=uncrnt_m_to_radians(1.0), longitude=uncrnt_m_to_radians(1.0), height=math.radians(45.0), pitch=math.radians(2.5), roll=math.radians(5.0), depth=0.05)
54
55 # Assume that we do have some level of calibration; uncertainty is relatively high since it's likely steel tap
56 vbi_off_u: Offset = Offset(x=0.05, y=0.05, z=0.05)
57 tpu = u.compute(observation, offset, vbi_meas_u, vbi_off_u)
58
59 # Assume that we have no level of calibration; uncertainty is therefore approximately the size of the ship!
60 vbi_off_u = Offset(x=5.0, y=2.0, z=3.0)
61 tpu = u.compute(observation, offset, vbi_meas_u, vbi_off_u)
62 report_uncertainty("WIBL Uncertainty (w/o Offset Measurement)", tpu)
63
64 if __name__ == "__main__":
65     main()
66
67 > OUTLINE
68 > TIMELINE
69
```

```
csbuncrnt > uncertainty.py > TotalPropagatedUncertainty > compute
1 from dataclasses import dataclass
2 import math
3 import numpy as np
4
5 @dataclass
6 class Measurement:
7     latitude: float
8     longitude: float
9     height: float
10    heading: float
11    pitch: float
12    roll: float
13    depth: float
14
15 @dataclass
16 class Offset:
17     x: float
18     y: float
19     z: float
20
21 class TotalPropagatedUncertainty:
22     # Computes the total propagated uncertainty of the georeferencing equation found in georeference.py by multipli
23     # Jacobian with the sensor covariance matrix
24     def compute(self, meas: Measurement, off: Offset, measu: Measurement, offu: Offset):
25         # The code below is set up to be evaluated as quickly as possible, so it's pretty messy and not easily rea
26         # In order to avoid having to pass lots of variables directly, they're structured in a couple of data clas
27         # but we need to unpack that in order to avoid having to modify the expanded code below.
28         latitude: float = meas.latitude
29         longitude: float = meas.longitude
30         ellipsoidalHeight: float = meas.height
31         heading: float = meas.heading
32         pitch: float = meas.pitch
33         roll: float = meas.roll
34         depth: float = meas.depth
35
36         x_offset: float = off.x
37         y_offset: float = off.y
38         z_offset: float = off.z
39
40         latitudeSigma: float = measu.latitude
41         longitudeSigma: float = measu.longitude
42         ellipsoidalHeightSigma: float = measu.height
43         headingSigma: float = measu.heading
44         pitchSigma: float = measu.pitch
45         rollSigma: float = measu.roll
46         depthSigma: float = measu.depth
47
48         xOffsetSigma: float = offu.x
49         yOffsetSigma: float = offu.y
50         zOffsetSigma: float = offu.z
51
52         return np.array([
53             ((ellipsoidalHeightSigma**2)*(math.cos(latitude)**2)*(math.cos(longitude)**2) + ((math.cos(latitude)*math
54             ((math.cos(roll)*math.sin(heading)*math.sin(pitch) - math.cos(heading)*math.sin(roll))*math.cos(longitud
55             ((ellipsoidalHeight + 6.3781370000000006/math.sqrt(-0.00669437999014140*(math.sin(latitude)**2) + 1))*mat
56             ((ellipsoidalHeight + 6.3781370000000006/math.sqrt(-0.00669437999014140*(math.sin(latitude)**2) + 1))*mat
57             y_offset_z,lati
```




IHO

CSB Tools Workshop Development Day

International
Hydrographic
Organization

The screenshot displays the OpenVBI Workflow Tool interface. The main window has three tabs: 'Inputs', 'Results', and 'Failed Files'. The 'Inputs' tab is active, showing fields for 'Input Directory' (Extras/OpenVBI/ExampleData), 'Output Directory' (/Users/brc/temp), and a 'Workflow' section with dropdowns for 'Loader' (WIBL), 'Writer' (DCDB GeoJson), and 'Depth Message' (Depth (NMEA2000)). A 'Metadata File' field contains 'tDay/rukuwai2-metadata.json'. A red arrow points from the 'Create' button in the 'Metadata File' section to the 'trustedNode' configuration window.

The 'trustedNode' configuration window is open, showing the following fields:

trustedNode	
providerOrganizationName	UNHJHC
providerEmail	wibl@ccom.unh.edu
uniqueVesselID	
convention	GeoJSON CSB 3.1
dataLicense	CC0 1.0
providerLogger	WIBL
providerLoggerVersion	1.3/1.1.0/1.0.1
navigationCRS	
verticalReferenceOfDepth	Transducer
vesselPositionReferencePoint	GNSS

Below the 'trustedNode' section is the 'platform' section:

platform	
type	Research vessel
name	Rukuwai II
length	
IDType	MMSI
IDNumber	512000771
sensors	
soundSpeedDocumented	
positionOffsetsDocumented	
dataProcessed	
contributorComments	
uniqueID	

At the bottom of the 'trustedNode' window is the 'Export' section:

Export	
Filename	velopmentDay/rukuwai2.json Choose...

Below the 'Export' section is the 'Actions' section:

Actions	
Validate	Export Quit



IHO

International
Hydrographic
Organization

CSB Tools Workshop Development Day

```
openvbi > adaptors > csb_rest_api_adapter.py > CSBRestApiLoader > /workspaces/OpenVBI/tests/unit/test_processing.py
1 import os
2 from pathlib import Path
3
4 import requests
5 from openvbi.adaptors.dcdb import CSVLoader
6 from openvbi.core.observations import Dataset
7 from openvbi.adaptors import Loader
8
9 class CSBRestApiLoader(Loader):
10     def __init__(self, output_path: Path, query_result_loader: Loader = CSVLoader(), default_bucket_path: str = 'https://noaa-dcdb-bathymetry-pds.s3.amazonaws.com/csb/csv') -> None:
11         self._query_result_loader = query_result_loader
12         self._output_path = output_path
13         self._bucket_path = resolve_bucket_path(default_bucket_path)
14
15     def suffix(self) -> str:
16         return '.csv'
17
18     def load(self, filename: str) -> Dataset:
19         object_path = get_object_path(self._bucket_path, filename)
20         file_path = download_object(object_path, filename, self._output_path)
21         return self._query_result_loader.load(file_path)
22
23     def resolve_bucket_path(default_bucket_path: str) -> str:
24         bucket_path = os.environ.get('OPENVBI_S3_BUCKET')
25         if bucket_path is None:
26             return default_bucket_path
27         return bucket_path
28
29     def get_object_path(bucket_path: str, filename: str) -> str:
30         file_name_without_extension: str = filename.replace(".tar.gz", "")
31         time_code = file_name_without_extension.split("_", 2)[0]
32         year = time_code[0:4]
33         month = time_code[4:6]
34         day = time_code[6:8]
35         return f"{bucket_path}/{year}/{month}/{day}/{file_name_without_extension}_pointData.csv"
36
37     def download_object(object_path: str, filename: str, output_path: Path) -> Path:
38         if output_path.exists() and output_path.is_dir():
39             print(f"{output_path} is valid")
40         else:
41             print(f"{output_path} is not a valid path")
42
43         with requests.get(object_path, stream=True) as response:
44             status_code = response.status_code
45             if status_code != 200:
46                 raise RuntimeError(f"object request status did not return 200, returned {status_code}")
47
48             with open(output_path.joinpath(filename), 'w') as output_file:
49                 for line in response.iter_lines():
50                     if line:
51                         output_file.write(line)
52
53         return output_path
54
55
56
```



IHO

CSB Tools Workshop Development Day

International
Hydrographic
Organization

CCOMJHC / OpenVBI

Search: Type / to search

Code Issues 8 Pull requests Discussions Actions Projects Wiki Security

Outlier detection/Data filtering/Cleaning tool #17

ecarle-metservice started this conversation in Ideas

ecarle-metservice 2 hours ago

Automated tool to filter / clean CSB datasets

Data issues to address

1. *Point Outliers* - depths are erroneously too large/too small relative to neighbouring data points
2. *Transect Outliers* - a collection/transect of points are all too large/too small relative to other data/transects
3. *Stalled logger* - depths are recorded as single erroneous value for an extended period
4. *Oversampling* - remove multiple repeated depth readings that are at the same / nearby location

Potential Cleaning Processes

Note this should be modular, you can decide which of the processes to run

- *Basic Checks*: filter depths/lat lon/time out of some defined acceptable range
- *Speed over ground checker*: compute approximate speed from timestamp/lat/lon and remove points w/speed > some kt
- *Point Outlier cleaning*: use point cloud detection tools, e.g. [Open3D](#) to filter outliers
- *Stalled logger*: if point and n nearest neighbours are same value/close to same value, then remove them
- *Reference dataset comparison*: compare dataset with some reference bathymetry (could be point, grid, or chart data). Compare dataset points with overlapping/"nearby" reference points/grid cells. Remove points which have disparity larger than some threshold

Category: Ideas

Labels: None yet

1 participant

Notifications: [Subscribe](#)

You're not receiving notifications from this thread.

Lock conversation

Transfer this discussion

Pin discussion

Pin discussion to Ideas

Create issue from discussion

Delete discussion

1 comment

ecarle-metservice 2 hours ago [Author](#)

Motivated by [#2 \(comment\)](#)

CCOMJHC / OpenVBI

Search: Type / to search

Code Issues 8 Pull requests Discussions Actions Projects

is:issue state:open

Labels Milestones New issue

Open 8 Closed 1

Author Labels Projects Milestones

- ☐ [Rapid feedback output products](#) [enhancement](#) [Feature](#) #15 · brian-r-calder opened last week
- ☐ [Uncertainty Estimation](#) [enhancement](#) [Feature](#) #12 · brian-r-calder opened last week
- ☐ [DCDB Data Discovery and Recovery](#) [enhancement](#) [Feature](#) #11 · brian-r-calder opened last week
- ☐ [Outlier rejection algorithms](#) [enhancement](#) [Feature](#) #10 · brian-r-calder opened last week
- ☐ [Vertical correctors \(water level and observation bias\)](#) [enhancement](#) [Feature](#) #9 · brian-r-calder opened last week
- ☐ [Workflow Management Tool](#) [enhancement](#) [Feature](#) #8 · brian-r-calder opened last week
- ☐ [Metadata Creation Tool](#) [enhancement](#) [Feature](#) #7 · brian-r-calder opened last week
- ☐ [Add data files for examples](#) #4 · heathhenley opened on Feb 13



IHO

Planned Activities

International
Hydrographic
Organization



- Updates to code in OpenVBI as consequence of CSB Tools Workshop development day (see separate report)



- Architecture review of OpenVBI base



- Capture of development ideas from CSB Tools Workshop
- Discussion with (former) Work Item F members to update task list for next year (at CSBWG16)



IHO

International
Hydrographic
Organization

“Official” Work Item H Tasks



H-1	Review, recommend, and document the data flow of standard processing stages for data capture	M	Present to CSBWG for discussion	2023	Dec 2024	O	CCOM/B Calder		
H-2	Review current metadata structure for CSB data for completeness, encoding methods, validation mechanisms, and support for end-user database mapping	M	Recommend modifications	2023	March 2025	P	CCOM/B Calder		
H-3	Consider potential extensions of the current <u>GeoJSON</u>	M	Recommend modifications	2023	Dec 2024	O	CCOM/B Calder		



H-4	Investigate possible extensions to the DCDB S3 data store and OGC services	H	Consolidate suggestions for future development at DCDB.	2023	May 2024	C	CCOM/B Calder		
H-5	Establish recommendations for one or more use cases of CSB, and associated guidelines for data products	M	Collaborate with other working groups preparing specialized products (e.g., for HO use)	2023	Ongoing	O	CCOM/B Calder		



IHO

Requests to CSBWG

International
Hydrographic
Organization

- Note the information provided in the written report & here
- Provide feedback on tasks, development direction as necessary