The SONAR-netCDF4 convention for sonar data, version 2.0

List of authors

Version 2.0

Table of Contents

Colophon

International Council for the Exploration of the Sea
Conseil International pour l'Exploration de la Mer

H. C. Andersens Boulevard 44-46
DK-1553 Copenhagen V
Denmark
Telephone (+45) 33 38 67 00
Telefax (+45) 33 93 42 15
www.ices.dk
info@ices.dk

ICES boilerplate stuff goes here.

Foreword

This report documents a convention for the storage of sonar data in netCDF4-formatted computer files. The intention is to provide a well-founded convention that is supported by multiple sonar systems and multiple sonar analysis software packages, with the aim of facilitating the use of sonar data for research and survey purposes. The name of this convention is SONAR-netCDF4.

This document was originally developed by the Topic Group on Defining a data format for omnidirectional fisheries sonar, part of ICES Working Group on Fisheries Acoustics, Science, and Technology (WGFAST). The current version was developed by the Open Data subgroup of WGFAST.

Chapter 1. Introduction

Sonars have long been used to study and understand fisheries and the aquatic environment, but only recently have they directly provided digital data for quantitative analysis. Each manufacturer typically provides such data in a proprietary, but usually open, file format specific to their sonar systems. This hinders the effective use and exchange of such data by requiring the development and maintenance of file-reading software for multiple analysis programs and multiple sonars.

This document presents a convention for the storage and exchange of fisheries sonar data, with an initial focus on omnisonars. It is sufficiently generic and flexible to contain all foreseeable types of fisheries sonar data, along with necessary metadata. It also serves as a statement of the minimum set of data and metadata required to use omnisonar data in a quantitative manner.

The term sonar is used in this convention to mean all underwater acoustic systems that actively transmit sound and then listen for echoes from that transmission.

1.1. Background

Many purpose-built file formats exist to store and exchange data from scientific and industrial equipment. Formats have been created by sonar and echosounder manufacturers; in addition, more generic acoustic data formats such as the Generic Water Column (GWC) format (Gee *et al.*, 2012), the eXtended Triton Format (XTF; Triton (2013)), and the HydroACoustics (HAC) format (McQuinn *et al.*, 2005) have also been created. These formats store a time-ordered sequence of datagrams, making it easy to append new data. Other data, such as geographical position, are typically interleaved into the sequence of datagrams. However, this is not optimal for analysis purposes when data are viewed and analysed as a set of pings and metadata from a time-period or spatial grouping. In addition, efficient random-read access to individual pings is not possible unless an index is available or created.

The work presented here does not create yet another sonar-specific file format; the knowledge required is not within the expertise of this group. However, the group does have the expertise to specify what data must be stored to allow for unambiguous use of backscatter data during quantitative analysis.

Accordingly, an existing file-format definition has been utilized and what should be stored is specified. The requirements for such a file format definition were:

- ability to adequately represent the content and structure of sonar data and associated metadata;
- standardized, open-file format;
- fast random access to data stored in the file;
- ability to store multiple types of data (e.g. position and backscatter);
- ability to store metadata (e.g. sonar settings);
- freely available and open libraries to read the data files into programming languages and technical computing environments (e.g. Java, C, C++, Python, Matlab, and R);
- reliable and space-efficient format for data exchange and storage;

- self-describing file format;
- backwards compatibility upon modification of the file contents specification (i.e. old software/tools can still read relevant parts of a newer version);
- computer platform-independent;
- long-term support and extensive use in other scientific fields;
- support for very large datasets.

Other scientific communities that collect and produce voluminous amounts of data have addressed similar needs, resulting in the Hierarchical Data Format (The HDF Group, 2017), currently at version 5 (HDF5). This is the only file format that meets all the listed requirements and is utilized here for sonar data. There are two realizations of the HDF5 file format: (i) HDF5 itself and (ii) netCDF4 (Unidata, 2017), which is a subset of HDF5. Both are sufficient, but netCDF4 is more widely used and has slightly wider language support and implementation diversity. Accordingly, netCDF4 has been chosen.

Using a well-supported file format has the significant benefit that many data exploration, query, extraction, and analysis tools already read such files. This eliminates the need to develop and maintain sonar-specific file-reading software and facilitates the use of existing tools for data management, distribution, and analysis.

An attractive feature of netCDF4 is the ease and transparency with which data can be added to an existing netCDF4 file. For example, processed data and how it was obtained can be included in the same file as the raw data. In addition, netCDF4 files can be very large while still providing fast access to data subsets. This allows a single SONAR-netCDF4 file to contain data from a transect or an entire survey. These features would simplify data management, improve traceability and long-term storage of analysis, enhance the sharing of processed datasets, and facilitate analysis of large disparate datasets. In general, any amount of additional data can be stored in the files without affecting the ability to use the data specified by the SONAR-netCDF4 convention.

A distinction is commonly made between file formats designed for archiving data in the original form, formats designed for storing partially or fully processed data, and formats for data exchange (Jackson *et al.*, 2014). The SONAR-netCDF4 convention is intended to be suitable for all of these.

Version 1 of this convention specified how to store backscatter amplitude data from omnisonars. Version 2 now provides for the storage of data from other acoustic equipment, including echosounders, multi-beam sonars, and Acoustic Doppler Current Profilers (ADCP). It also has proposed implementations for the storage of derived data, such as bottom and school detections, categorization of backscatter, and integrated backscatter at multiple resolutions.

1.2. Versioning

This document and the convention that it describes will change over time to implement enhancements and to correct errors and omissions. To accommodate this, the SONAR-netCDF4 convention always requires a version number. This document has a separate version number, allowing for revised versions of the document to describe the same version of the convention.

The convention version number will always be included in the title of the document. The document

version number will always be found in Chapter 7.

Chapter 2. Use of the SONAR-netCDF4 convention in echo-integration processing

There are several processing steps that are common among the echo-integration postprocessing programs, and there is a need to ease the use of such programs interchangeably. This section describes a few standard processing steps and outlines how the convention is intended to be used to facilitate this. Note that the steps may be different between programs, and the convention only recommends how to store the data for the steps that are most common. We do not address how and in which order to process the data.

Exchange of information between different parts in the processing chain is increasingly important. This is particularly relevant for deep learning and machine learning frameworks, which have hardware requirements that are often different to the computers that are used for traditional desktop applications. Processing tools that run on autonomous platforms are also a need, which often require a different suite of software solutions, typically tailored to low power situations.

Typical steps in the postprocessing chain of sonar data include noise removal, seabed detection, target classification, and echo-integration (Figure 1). Here we outline the different groups in the convention that can support the input and output from these steps. If a processing software step can read the data, process the information and write back to the data using the convention, other software can pick it up and process it further.

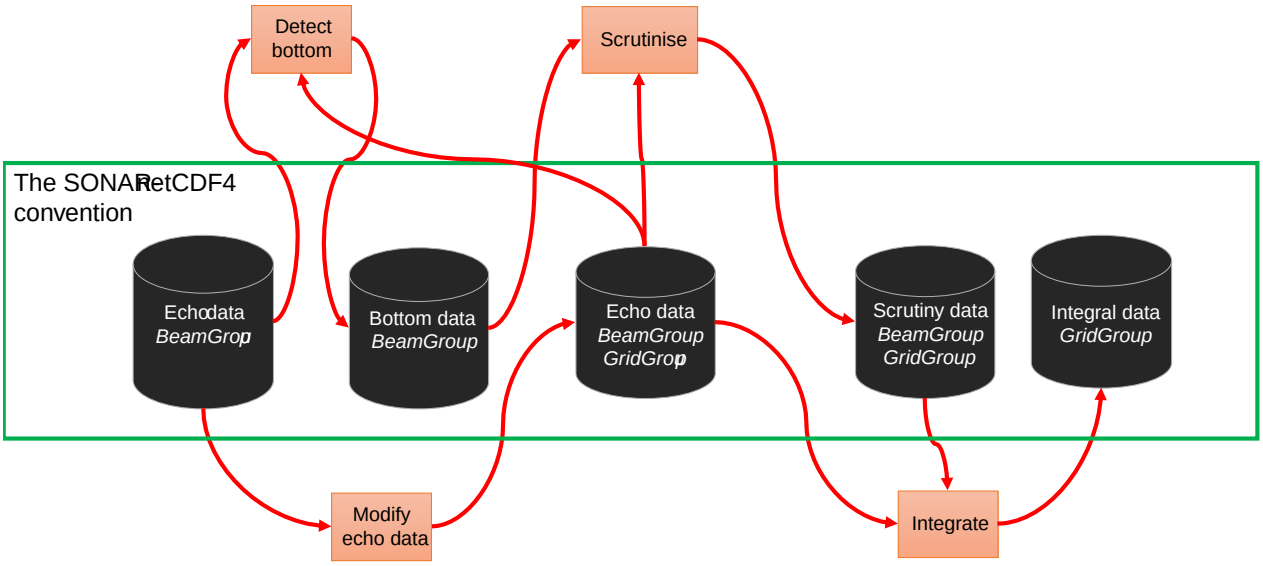

Figure 1. Typical steps in an echo-integration processing chain for water column data. The two main groups may be used for several steps depending on the objective, and the `GridGroup` may be seen as a projection of the `BeamGroup` and the conversion to `GridGroup` is typically lossy. The output from the sonars are typically the `BeamGroup` and the report format are typically the `GridGroup`.

2.1. Typical processing steps and the associated data models

Two main data models are available in the convention: `Beam_group` and `Gridded`. The `Beam_group` allows for different sample rates, different ping rates, etc, for each channel. It is a flexible model and able to cope with all sonar systems. The `Gridded` is a storage structure for multi-channel data that has been re-gridded or has homogeneous ping times and range bins across all channels (i.e., frequencies).

Depending on the type of processing, both `Beam_group` and `Gridded` could be used as input and output.

2.1.1. Noise removal

to come

2.1.2. Seabed detection

Range to the seabed is needed when echo-integrating biological backscatter close to the seabed. Different algorithms have been developed and the recommendation is to store the range to the seabed in the `Beam_group`.

2.1.3. Labeling

Labeling acoustic data, i.e. scrutinizing and allocating the acoustic energy to categories, is traditionally achieved by manually assigning backscatter. A data structure to contain such labels is detailed in Section 3.9.9.

An earlier overview of how different echo-integration software stores this information can be found here: Label overview

2.1.4. Machine learning

Convolutional neural networks and other machine learning methods are suitable for processing sonar data, but it is typically more convenient to map the data to a common range, time and frequency grid before passing it to a deep learning framework. The use of `Gridded` on a high resolution grid should be used for this purpose.

2.1.5. Integrated backscatter

The output from the echo-integration process is typically further processed to yield distribution and abundance estimates of selected species. It is recommended to use the `Gridded` for this purpose, where the data is gridded typically on a coarser grid that that used for the machine learning objective.

Chapter 3. The SONAR-netCDF4 convention

3.1. Introduction

NetCDF4 is a data model, application programming interface (API) library, and file format for storing and managing data, developed and maintained by Unidata. Unidata is part of the US University Corporation for Atmospheric Research (UCAR) Community Programs (UCP) and funded primarily by the US National Science Foundation.

SONAR-netCDF4 is a data and metadata convention for the storage of data from active sonars in netCDF4 formatted files. SONAR-netCDF4 consists primarily of a naming convention and a data structure within the netCDF4 data model.

Datasets can be added to SONAR-netCDF4 files if they do not conflict with the SONAR-netCDF4 datasets. If such additions are potentially of use to other users of the file format, it is recommended that they be proposed for inclusion in this or additional convention specifications.

Each SONAR-netCDF4 file is intended to store data from one sonar mounted on one platform. The storage of data from multiple sonars and multiple platforms in one SONAR-netCDF4 file is not in the scope of the convention.

A design principle of SONAR-netCDF4 has been to focus on describing the acoustic backscatter data, not the overall purpose and context of why the data were collected. Such broader metadata are better stored in separate metadata systems and schema (e.g. ISO (2014); ICES (2016)).

3.2. Hierarchical structure

NetCDF4 has two main organizational concepts: (i) the variable, which can contain a variety of data structures, and (ii) groups, being a collection of variables. Groups and variables can have attributes attached to them. Groups can be arranged into a hierarchy. SONAR-netCDF4 divides sonar data into a set of hiearchical netCDF4 groups:

1. / – The top-level or root of the hierarchy and contains metadata about the SONAR-netCDF4 file format;
    a. **Annotation** – contains time-stamped annotations;
    b. **Environment** – contains information relevant to acoustic propagation through water;
    c. **Platform** – contains information about the platform on which the sonar is installed and information about each sensor, e.g. its relative position to the platform frame of reference.
    d. **Provenance** – contains metadata about how the SONAR-netCDF4 version of the data were obtained and processed;
    e. **Sonar** – contains the main data models in the convention; groups under Sonar are used for storing the data output from the sonars, as well as processed sonar data and interpretation masks;
        i. **Beam_groupX** – contains ping based backscatter; the group supports multiple sensors of the same type, e.g. multiple frequencies frequencies;

ii. **GriddedX** – contains gridded backscatter data; the grid is similar across multiple frequencies and can be used for fine scale grids as well as coarser resolution typically as a basis for the reports;

iii. **InterpretationX** – contains the interpretation regions; the intepretation mask is independent from the backscatter groups;

f. **Vendor specific** – contains vendor-specific information about the sonar and the data.

The 'X' suffix on some group names indicates that there can be multiple of these type of groups and that the true group name will have a trailing number.

These groups contain variables and variable attributes with prescribed names and contents.

3.3. Obligations and missing data

Some variables and attributes in SONAR-netCD4 are mandatory; these form the minimal set of data required to quantitatively use backscattering amplitude data. The remaining variables and attributes have various levels of optionality and provide enhanced context and information about the sonar data. The obligations are:

1. **M**: mandatory
2. **MA**: mandatory if applicable or available
3. **R**: recommended
4. **O**: optional

Any non-mandatory variables can be absent from a SONAR-netCDF4 file. If a variable is mandatory, it must be present and must contain data. The set of mandatory variables and attributes has been chosen so that sonar systems can directly generate SONAR-netCDF4-conforming files without needing survey, experiment, or cruise-specific data.

The _FillValue attribute should be used to indicate missing data in variables. For floating point values, the IEEE 754 not-a-number (NaN) is the preferred fill value as this is convenient for commonly-used analysis packages (e.g. Python, Matlab, R).

3.4. Metadata and authorities

The fisheries acoustics community has developed a metadata convention for processed acoustic data (ICES, 2016). Where relevant, attribute and variable names have been reused from this convention. The NetCDF Climate and Forecast (CF) Metadata Conventions (Eaton *et al.*, 2017) have been used where sensible (the efficient storage of unprocessed active sonar data has not been a design goal of the CF convention) along with the ESIP Attribute Convention for Data Discovery (ACDD). Terms and concepts from other metadata conventions have also been used where appropriate.

3.5. Units

All relevant variables and attributes in SONAR-netCDF4 files are required to have a textual netCDF4

attribute with the name “units” that specifies the units. The International System of Units (SI) is used. For simplicity, the data format mandates the use of particular units and their textual form, as per the definitions and conventions of the UDUNITS-2 package (UCAR, 2014).

Decibels are commonly used in underwater acoustic quantities, but UDUNITS does not currently include such a unit. However, as the CF convention does permit decibel (with symbols: “dB”, “db”, and “dbel”), it is used in the SONAR-netCDF4 convention. Note that reference values commonly used with decibels should not be part of the units attribute, but rather included in the variable description (see Nedelec *et al.* (2021), §11.4 for an explanation of this).

Numeric variables that are dimensionless should have a units attribute of "1".

3.5.1. Time units

Even though NetCDF and the UDUNITS-2 part of the CF conventions allow for the specification any valid time unit, to prevent any extra complication in software that reads the SONAR-netCDF4 files, only two time units are supported in this convention.

- nanoseconds since 1601-01-01 00:00:00Z
- nanoseconds since 1970-01-01 00:00:00Z

The choice of nanoseconds is because sonar system timestamp precision can be less than one millisecond. Other forms of time units, such as 100 nanoseconds, are not well supported by the various tools that read CF-formatted time units, and nanoseconds units is retained.

The year 1601 is used because it is the epoch year unit used in Simrad .raw files and Windows32/64 systems.

The year 1970 is used because it is the epoch time widely used in Unix and most programming languages.

3.6. Datatypes

Each item has a suggested datatype, chosen to have sufficient range and precision to contain the expected data. Alternative datatypes can be used if necessary, but are discouraged. The “string” type should contain text in the UTF-8 encoding and should be treated as case-sensitive during any comparisons. Enumerated datatypes are used for some of the controlled vocabularies.

3.7. Vocabularies

The contents of some variables and attributes are restricted to defined vocabularies. These are listed or referenced where required. Desired additions to the vocabularies should be proposed to WGFAST for incorporation into this document. Some of the vocabularies have been represented as netCDF4 enumeration data types and some using the CF flag_values convention.

3.8. File-naming convention

SONAR-netCDF4 files should always end with a “.nc” suffix to indicate that they are a netCDF file. It

is recommended that the filename should sort alphanumerically into chronological order (e.g. date and time of the first ping in the file; thus: YYYYMMDD-HHMMSS.nc). This facilitates file management and use in analysis systems.

3.9. Groups

The top-level group (labelled "/" in netCDF4 terminology) contains metadata about the SONAR-netCDF4 file, represented as attributes in the group (Table 1).

Table 1. Description of the top-level group.

Description	Obligation	Comment
Group attributes		
:Conventions = "CF-1.7, SONAR-netCDF4-2.0, ACDD-1.3"	M	A comma-separated list of the conventions followed in the file. Include the SONAR-netCDF4 convention and version (e.g. ”SONAR-netCDF4-2.0”) and the relevant CF and ACDD conventions (e.g. ”CF-1.7” and ”ACDD-1.3”).
:date_created	M	Timestamp of file creation in ISO8601:2004 extended format, including the time zone (e.g. 2017-05-06T20:21:35Z).
:keywords	M	A comma-separated list of key words and/or phrases. For direct sonar-generated files, this should at least include the type of sonar.
:license	O	Either enter the URL to a standard or specific license, enter “Freely distributed” or “None”, or describe any restrictions to data access and distribution in free text.
:rights	O	Description of the usage rights of data in the file.
:sonar_convention_authority = "ICES"	M	Name of the organization managing and distributing the SONAR-netCDF4 convention. Currently ICES.
:sonar_convention_name = "SONAR-netCDF4"	M	Formal name of this convention (i.e. ”SONAR-netCDF4”).
:sonar_convention_version = "2.0"	M	SONAR-netCDF4 version number in the form “major.minor”, where major and minor are non-negative integers.
:summary	M	A paragraph describing the dataset, analogous to an abstract for a paper. For direct sonar-generated files, this can be blank.
:title	M	A short phrase or sentence describing the dataset. For direct sonar-generated files, this can be as simple as "Files generated by the XYZ sonar".

3.9.1. Annotation group

The annotation group contains timestamped annotations with optional identification code. Annotations are typically textual notes entered by users relevant to the data at a particular time. Some sonar systems provide an interface for creating manual and programmatic annotations. The netCDF4 group name is /**Annotation** and is described in Table 2.

Table 2. Description of the annotation group.

Description	Obligation	Comment
Dimensions		
time = unlimited		Can be of fixed or unlimited length, as appropriate.
Coordinate variables		
uint64 time(time)	MA	
:axis = "T"		
:calendar = "gregorian"		
:long_name = "Timestamps of annotations"		
:standard_name = "time"		
:units = "nanoseconds since 1970-01-01 00:00:00Z" or "nanoseconds since 1601-01-01 00:00:00Z"		
Variables		
string annotation_category(time)	O	Optional category for the annotation, for use in grouping annotation types.
:long_name = "Annotation category"		
string annotation_text(time)	MA	
:long_name = "Annotation text"		

3.9.2. Environment group

The environment group contains information on environmental conditions, especially the speed of sound in water and acoustic absorption. The netCDF4 group name is /**Environment** and is described in Table 3. Sound speed, absorption, and current profiles can also be stored in this group, as can profile measurements of salinity, temperature, and pressure. Such profile data should use the NCEI NetCDF “profile” template, v2.0 or greater.

Table 3. Description of the environment group.

Description	Obligation	Comment
Dimensions		
frequency = unlimited		Can be of fixed or unlimited length, as appropriate.
Coordinate variables		
float frequency(frequency)	M	
:long_name = "Acoustic frequency"		
:standard_name = "sound_frequency"		
:units = "Hz"		
float :valid_min = 0.0		
Variables		
float absorption_indicative(frequency)	M	Indicative absorption values used to calculate the time-varied-gain (TVG), in the absence of more detailed data.
:long_name = "Indicative acoustic absorption"		
:units = "dB/m"		
float :valid_min = 0.0		
float sound_speed_indicative	M	Mean sound speed in water used to calculate echo range, in the absence of more detailed sound-speed data.
:long_name = "Indicative sound speed"		
:standard_name = "speed_of_sound_in_sea_water"		
:units = "m/s"		
float :valid_min = 0.0		

3.9.3. Platform group

This group contains information about the sonar platform (e.g. ship). The netCDF4 group name is /**Platform** and is described in Table 4. Optionally, subgroups of /Platform can be used to store data from individual instruments that provide measurements about the platform, such as position, attitude, etc (note that if the measurements are about the environment around the platform they should be stored in the /Environment group). This structure must be used to store all instrument data.

Each platform can contain several instruments of the same type, such as multiple position sensors. Each instrument type is defined as a subgroup of the /Platform group and data for each instrument are stored in a subgroup of the instrument group. Subgroup names for common instrument types are given in Table 5 and how to store data from those instruments are also cross-referenced in Table 5. For example, data from individual position and attitude instruments are stored as subgroups of the /Platform/Position and /Platform/Attitude subgroups. The names of those subgroups must match the names of the instrument's serial number or identifier as defined in the MRU_ids and position_ids variables in the /Platform group (Table 4). Additional instrument type subgroups can be created as required.

Each instrument group can also store unprocessed and unparsed sensor data. The data format for unprocessed and unparsed data is not prescribed, but could, for example, be NMEA-style text and/or numeric values. These subgroups must have one attribute called "description" that provides a short description of the contents. Other attributes can be added as desired. The variables under the subgroup should have appropriate names and an attribute that gives the units, where appropriate.

As a general example, parsed data from a GPS with id *Garmin1234* would be stored in /Platform/Position/Garmin1234 and if NMEA data from that GPS were also stored, it would be in /Platform/Position/Garmin1234/NMEA.

The coordinate system convention used for the /Platform group variables is detailed in Chapter 5.

Table 4. Description of the platform group.

Description	Obligation	Comment
Group attributes		
:platform_code_ICES	O	Platform code from the ICES SHIPC vocabulary (http://vocab.ices.dk/services/pox/GetCodeList/SHIPC).
:platform_name	O	Platform name of which the sonar is a part.
:platform_type	O	Platform type that the sonar is part of. Use the description field from the ICES ”Platform Class” vocabulary (http://vocab.ices.dk/services/pox/GetCodeList/Platform%20Class).
Types		
byte enum transducer_type_t {receive_only = 0, transmit_only = 1, monostatic = 3 }		Transducer function - transmit only, receive only or both (monostatic)
Dimensions		
transducer	M	Transducer count
position	M	position sensor count
MRU	M	MRU attitude sensor count
Variables		
float MRU_offset_x(MRU)	R	*x*-axis component of the vector from the platform coordinate system origin to the motion reference unit origin.
:long_name = "Distance along the *x*-axis from the platform coordinate system origin to the motion reference unit sensor origin"		
:units = "m"		
float MRU_offset_y(MRU)	R	*y*-axis component of the vector from the platform coordinate system origin to the motion reference unit origin.
:long_name = "Distance along the *y*-axis from the platform coordinate system origin to the motion reference unit sensor origin"		
:units = "m"		
float MRU_offset_z(MRU)	R	*z*-axis component of the vector from the platform coordinate system origin to the motion reference unit origin.
:long_name = "Distance along the *z*-axis from the platform coordinate system origin to the motion reference unit sensor origin"		
:units = "m"		
float MRU_rotation_x(MRU)	R	Extrinsic angular rotation about the *x*-axis from the platform zero angle to the MRU zero angle.
:long_name = "Extrinsic rotation about the *x*-axis from the platform to MRU coordinate systems"		
:units = "arc_degree"		
float :valid_range = −180.0, 180.0		
float MRU_rotation_y(MRU)	R	Extrinsic angular rotation about the *y*-axis from the platform zero angle to the MRU zero angle.
:long_name = "Extrinsic rotation about the *y*-axis from the platform to MRU coordinate systems"		

Description	Obligation	Comment
:units = "arc_degree"		
float :valid_range = −180.0, 180.0		
float MRU_rotation_z(MRU)	R	Extrinsic angular rotation about the *z*-axis from the platform zero angle to the MRU zero angle.
:long_name = "Extrinsic rotation about the *z*-axis from the platform to MRU coordinate systems"		
:units = "arc_degree"		
float :valid_range = −180.0, 180.0		
string MRU_ids(MRU)	MA	MRU serial number or identification name. Must be a valid netCDF4 group name.
string position_ids(position)	MA	Position sensor serial number or identification name. Must be a valid netCDF4 group name.
float position_offset_x(position)	R	Distance from the platform coordinate system origin to the latitude/longitude position origin along the *x*-axis.
:long_name = "Distance along the *x*-axis from the platform coordinate system origin to the latitude/longitude sensor origin"		
:units = "m"		
float position_offset_y(position)	R	Distance from the platform coordinate system origin to the latitude/longitude position origin along the *y*-axis.
:long_name = "Distance along the *y*-axis from the platform coordinate system origin to the latitude/longitude sensor origin"		
:units = "m"		
float position_offset_z(position)	R	Distance from the platform coordinate system origin to the latitude/longitude position origin along the *z*-axis.
:long_name = "Distance along the *z*-axis from the platform coordinate system origin to the latitude/longitude sensor origin"		
:units = "m"		
float transducer_offset_x(transducer)	R	Distance from the platform coordinate system origin to the transducer along the *x*-axis.
:long_name = "*x*-axis distance from the platform coordinate system origin to the sonar transducer"		
:units = "m"		
float transducer_offset_y(transducer)	R	Distance from the platform coordinate system origin to the transducer along the *y*-axis.
:long_name = "*y*-axis distance from the platform coordinate system origin to the sonar transducer"		
:units = "m"		
float transducer_offset_z(transducer)	R	Distance from the platform coordinate system origin to the transducer along the *z*-axis.
:long_name = "*z*-axis distance from the platform coordinate system origin to the sonar transducer"		
:units = "m"		
string transducer_ids(transducer)	MA	Transducer serial number or identification name
float transducer_rotation_x(transducer)	R	Extrinsic angular rotation about the *x*-axis from the transducer zero angle to the coordinate system origin zero angle.
float :valid_range = −180.0, 180.0		

Description	Obligation	Comment
:units = "arc_degree"		
:long_name = "Extrinsic rotation about the *x*-axis from the transducer to reference coordinate systems"		
float transducer_rotation_y(transducer)	R	Extrinsic angular rotation about the *y*-axis from the transducer zero angle to the coordinate system origin zero angle.
float :valid_range = −180.0, 180.0		
:units = "arc_degree"		
:long_name = "Extrinsic rotation about the *y*-axis from the transducer to reference coordinate systems"		
float transducer_rotation_z(transducer)	R	Extrinsic angular rotation about the *z*-axis from the transducer zero angle to the coordinate system origin zero angle.
float :valid_range = −180.0, 180.0		
:units = "arc_degree"		
:long_name = "Extrinsic rotation about the *z*-azis from the transducer to reference coordinate systems"		
transducer_type_t transducer_function(transducer)	M	The transducer function (that is, transmit_only, receive_only, or monostatic)
:long_name = "Transducer function (transmit_only, receive_only, monostatic)"		
float water_level	R	Distance from the origin of the platform coordinate system to the nominal water level measured along the *z*-axis of the platform coordinate system (positive values are below the origin). The distance between the nominal and actual water level is provided by vertical_offset.
:long_name = "Distance from the platform coordinate system origin to the nominal water level along the *z*-axis"		
:units = "m"		
Subgroups		
Positions	M	Suggested subgroup to store Position sensor data.
Attitudes	M	Suggested subgroup to store MRU sensor data.

Table 5. Suggested subgroup names for platform instruments.

Sensor or datagram	Subgroup name	Comment
Attitude sensors, MRU	Attitude	Use data structure described in Table 6
GPS sensors, Position	Position	Use data structure described in Table 7
Clock or time synchronisation datagrams	Clock	
NMEA telegrams	NMEA	Use data structure described in Table 8.

Table 6. Description of a platform attitude subgroup.

Description	Obligation	Comment
Group attributes		
string :description	O	System description
Dimensions		
time		can be fixed or unlimited length, as appropriate
Coordinate variables		
uint64 time(time)	M	Time from attitude sensor
:axis = "T"		
:calendar = "gregorian"		
:long_name = "Timestamps for attitude data"		
:standard_name = "time"		
:units = "nanoseconds since 1970-01-01 00:00:00Z" or "nanoseconds since 1601-01-01 00:00:00Z"		
Variables		
float heading(time)	MA	Platform heading. Measured clockwise from north.
:long_name = "Platform heading(true)"		
:standard_name = "platform_orientation"		
:units = "degrees_north"		
float :valid_range = 0.0, 360.0		
float heading_rate(time)	MA	Platform heading rate.
:long_name = "Platform heading rate"		
:units = "degree/s"		
float pitch(time)	M	Platform pitch. Positive values indicate a bow-up pitch.
:standard_name = "platform_pitch_angle"		
:units = "arc_degree"		
:long_name = "pitch angle"		
float :valid_range = −90.0, 90.0		
float pitch_rate(time)	O	Platform pitch rate
:standard_name = "platform_pitch_rate"		
:units = "degree/s"		
:long_name = "pitch rate"		

Description	Obligation	Comment
float roll(time)	M	Platform roll. Positive values indicate a roll to starboard.
:standard_name = "platform_roll_angle"		
:units = "arc_degree"		
:long_name = "roll angle"		
float roll_rate(time)	O	Platform roll rate
:standard_name = "platform_roll_rate"		
:units = "degree/s"		
:long_name = "roll rate"		
float vertical_offset(time)	M	Distance from the nominal water level to the actual water level measured along the _z_axis of the platform coordinate system (positive values are when the actual water level is below the nominal water level). For ships and similar, this is called heave, but the concept applies equally well to underwater vehicle depth.This offset is applied at the position given by (MRU_offset_x, MRU_offset_y, MRU_offset_z).
:units = "m"		
:long_name = "Platform vertical offset from nominal"		
Subgroups		
NMEA	O	Suggested subgroup to store raw NMEA data.

Table 7. Description of a platform position subgroup.

Description	Obligation	Comment
Group attributes		
string :description	O	System description
Dimensions		
time		can be fixed or unlimited length, as appropriate
Coordinate variables		
uint64 time(time)	M	Time from position sensor
:axis = "T"		
:calendar = "gregorian"		
:long_name = "Timestamps for position data"		
:standard_name = "time"		
:units = "nanoseconds since 1970-01-01 00:00:00Z" or "nanoseconds since 1601-01-01 00:00:00Z"		

Description	Obligation	Comment
:coordinates = "time latitude longitude"		
Variables		
double latitude(time)	M	Latitude of the platform reference point in WGS-84 reference system
double :valid_range = −90.0, 90.0		
:standard_name = "Platform latitude"		
:units = "degrees_north"		
:long_name = "latitude"		
:coordinates = "time latitude longitude"		
double longitude(time)	M	Longitude of the platform reference point in WGS-84 reference system
double :valid_range = −180.0, 180.0		
:standard_name = "Platform longitude"		
:units = "degrees_east"		
:long_name = "longitude"		
:coordinates = "time latitude longitude"		
float heading(time)	MA	Heading refers to the direction a platform is pointing. This may or may not be the direction that the platform actually travels, which is known as its course or track. Any difference between course and heading is due to the motion of the underlying medium, the air or water, or other effects like skidding or slipping. Heading is typically based on compass directions, so 0° (or 360°) indicates a direction toward true North, 90° indicates a direction toward true East, 180° is true South, and 270° is true West.
:standard_name = "platform_orientation"		
:units = "degree"		
:long_name = "Platform heading(true)"		
:coordinates = "time latitude longitude"		
float course_over_ground(time)	O	a platform course is the cardinal direction along which the platform is to be steered
:standard_name = "platform_course"		
:units = "m/s"		
:long_name = "degree"		
:coordinates = "time latitude longitude"		
float speed_over_ground(time)	MA	Speed is the magnitude of velocity. The platform speed with respect to ground is relative to the solid Earth beneath it, i.e. the sea floor for a ship.
:standard_name = "platform_speed_wrt_ground"		

Description	Obligation	Comment
:units = "m/s"		
:long_name = "speed_over_ground"		
float :valid_min = 0.0		
:coordinates = "time latitude longitude"		
float speed_relative(time)	MA	Platform speed relative to water.
:long_name = "Platform speed relative to water"		
:standard_name = "platform_speed_wrt_seawater"		
:units = "m/s"		
float :valid_min = 0.0		
:coordinates = "time latitude longitude"		
float height_above_reference_ellipsoid(time)	MA	Height of the platform reference point above the WGS-84 ellipsoid
:standard_name = "height_above_reference_ellipsoid"		
:units = "m"		
:long_name = "height_above_reference_ellipsoid"		
:coordinates = "time latitude longitude"		
float altitude(time)	MA	Altitude is the (geometric) height of the platform reference point above the reference WGS-84 geoid. The geoid is similar to mean sea level.
:standard_name = "altitude"		
:units = "m"		
:long_name = "altitude"		
:coordinates = "time latitude longitude"		
float distance(time)	O	Distance travelled by the platform from an arbitrary location.
:long_name = "Distance travelled by the platform"		
:units = "m"		
float :valid_min = 0.0		
:coordinates = "time latitude longitude"		
Subgroups		
NMEA	O	Suggested subgroup to store raw NMEA data.

Table 8. Suggested group for storing NMEA datagrams from marine instruments.

Description	Obligation	Comment
Group attributes		
:description = "All NMEA sensor datagrams"	M	Description of the subgroup contents.
Dimensions		
time		Can be fixed or unlimited length, as appropriate.
Coordinate variables		
uint64 time(time)	M	
:axis = "T"		
:calendar = "gregorian"		
:long_name = "Timestamps for NMEA datagrams"		
:standard_name = "time"		
:units = "nanoseconds since 1970-01-01 00:00:00Z" or "nanoseconds since 1601-01-01 00:00:00Z"		
Variables		
string NMEA_datagram(time)	O	Example of how to store NMEA datagrams.
:long_name = "NMEA datagram"		

3.9.4. Provenance group

The provenance group provides information on how the SONAR-netCDF4 version of the data were created and what processes were applied to this data. The netCDF4 group name is /**Provenance** and is described in Table 9.

Table 9. Description of the provenance group.

Description	Obligation	Comment
Group attributes		
:conversion_software_name	MA	Name of the software used to do the conversion.
:conversion_software_version	MA	Version of the software used to do the conversion.
:conversion_time	MA	Date and time of the start of the conversion process in ISO8601:2004 extended format, including time zone.
:history	R	Provides an audit trail for modifications to the original data. It should contain a separate line for each modification, with each line beginning with a timestamp and including user name, modification name and modification arguments. for example 2019-09-07T15:50+00Z (John Doe) File conversion by XXX software
Dimensions		
filenames = unlimited	MA	Can be of fixed or unlimited length, as appropriate.
Variables		
string source_filenames(filenames)	MA	Vector of datafile names that were used to generate the data in this SONAR-netCDF4 file.
:long_name = "Source filenames"		

3.9.5. Sonar group

The netCDF4 group name is /**Sonar** and contains three subgroups.

Beam group

This group contains ping based sonar backscatter data and associated metadata and is described in Table 10. The netCDF4 group name is **Beam_groupX**, where **X** is an integer.

Data from each beam mode (e.g. horizontal and vertical beam modes) are stored in subgroups under the /Sonar group (see Table 11). The form of the backscatter data can vary between different sonar systems. For example, some provide a complex-valued amplitude, while others provide a real- or integer-valued amplitude.

Subgroups under /Sonar each have a coordinate variable that contains ping timestamps. In some cases the coordinate variables in different subgroups contain the same data (such as when a sonar produces several types of beam data from each and every ping). To avoid duplication of timestamp data, a coordinate variable can be used across multiple subgroups. For organisational reasons, it is then recommended that such coordinate variables be located in the /Sonar group.

Table 10. Description of the sonar group.

Description	Obligation	Comment
Group attributes		
:sonar_manufacturer	R	Name of the sonar manufacturer.
:sonar_model	R	Name of the sonar model.
:sonar_serial_number	R	Sonar serial number.
:sonar_software_name	R	Sonar software name.
:sonar_software_version	R	Sonar software version.
:sonar_type = "omni-sonar"	M	Type of sonar, chosen from a defined vocabulary (currently only one value): ”omnisonar” (a sonar with a nominally omnidirectional mode).
Types		
byte enum beam_stabilisation_t {not_stabilised = 0, stabilised = 1}		Whether or not the beam direction is compensated for platform motion.
byte enum beam_t {single = 0, split_aperture_angles = 1, split_aperture_4_subbeams = 2, split_aperture_3_subbeams = 3, split_aperture_3_1_subbeams = 4}		Beam type. Split aperture indicates a beam that can detect the arrival angle of echoes, while single beam cannot.
byte enum conversion_equation_t {type_1 = 1, type_2 = 2, type_3 = 3, type_4 = 4, type_5 = 5}		The type of equation used to convert backscatter measurements into volume backscattering and target strength.
byte enum transmit_t {CW = 0, LFM = 1, HFM = 2}		Type of transmit pulse. CW = continuous wave – a pulse nominally of one frequency, LFM = linear frequency modulation – a pulse which varies from transmit_frequency_start to transmit_frequency_stop in a linear manner over the nominal pulse duration (transmit_duration_nominal), HFM = hyperbolic frequency modulation – a pulse which varies from transmit_frequency_start to transmit_frequency_stop in a hyperbolic manner over the nominal pulse duration (transmit_duration_nominal).
Subgroups		
Beam_group1	M	Example of a beam group. Include as many subgroups as necessary for different beam groups. Use unique group names, preferably of the form Beam_groupX where X is an integer.
Grid_group1	M	Example of a grid group. Include as many subgroups as necessary for different grid groups containing different beam groups resampled on a user defined regular grid. Use unique group names, preferably of the form Grid_groupX where X is an integer.

Table 11. Description of the beam subgroups of the sonar group.

Description	Obligation	Comment
Group attributes		

Description	Obligation	Comment
:beam_mode	M	Mode of the beam in this subgroup, taken from the defined vocabulary of: “vertical” (a set of beams that form a vertical slice through the water), ”horizontal” (a set of beams that form a nominally horizontal plane through the water), and ”inspection” (a set of beams with arbitrary pointing directions).
conversion_equation_t :conversion_equation_type	M	Type of equation used to convert backscatter measurements into volume backscattering strength and target strength.
int :preferred_MRU	MA	Index of the MRU sensor to use by default. If the sensor used can be dynamically changed, refer to the variable active_MRU. Index matches the ones used in the Platform MRU sensors variables.
int :preferred_position	MA	Index of the position sensor to use by default. If the sensor used can be dynamically changed, refer to the variable active_position_sensor. Index matches the ones used in the Platform position sensors variables.
Types		
float(*) sample_t		Variable length vector used to store ragged arrays of backscatter data. Data type can be varied to suit data storage needs.
float(*) angle_t		Variable length vector used to store ragged arrays of split-aperture angles. Data type can varied to suit data storage needs.
float(*) pulse_t		Variable length vector used to store arrays of modeled transmit pulse samples.
Dimensions		
beam		The number of beams in this beam group.
subbeam		The number of sub-beams in these beams.
ping_time = unlimited		Can be of fixed or unlimited length, as appropriate.
tx_beam		The number of transmit beams in this beam group
frequency		The number of frequencies during the calibration exercise of the FM mode.
Coordinate variables		
string beam(beam)	M	Beam name (or number or identification code).
:long_name = "Beam name"		
uint64 ping_time(ping_time)	M	Timestamp at which each ping occurred.
:axis = "T"		
:calendar = "gregorian"		
:long_name = "Time-stamp of each ping"		
:standard_name = "time"		
:units = "nanoseconds since 1970-01-01 00:00:00Z" or "nanoseconds since 1601-01-01 00:00:00Z"		
:coordinates = "ping_time platform_latitude platform_longitude"		
float frequency(frequency)	MA	Frequencies for which transducer gain values have been estimated during the calibration exercise.
:long_name = "Calibration gain frequencies"		

Description	Obligation	Comment
:units = "Hz"		
float :valid_min = 0.0		
Variables		
sample_t backscatter_i(ping_time, beam, subbeam)	MA	Imaginary part of backscatter measurements (or for type 5 equations, TS values). Each element in the 3D matrix is a variable length vector (of type sample_t) that contains the samples for that beam, ping time, and optionally subbeam.
:long_name = "Raw backscatter measurements (imaginary part)"		
:units = "as appropriate"		Use units appropriate for the data.
int sample_count(ping_time, beam, subbeam)	O	The number of samples in each subbeam/beam/ping in the backscatter_r and backscatter_i variables. This value is not essential, but software that reads the backscatter_r and backscatter_i variables can use it to significantly improve data loading times.
:long_name = "Number of samples per ping in each beam, and optionally subbeam"		
:units = "1"		
int :valid_min = 0		
sample_t backscatter_r(ping_time, beam, subbeam)	M	Real part or amplitude or power of backscatter measurements. Each element in the 3D matrix is a variable length vector (of type sample_t) that contains the samples for that beam, ping time, and optionally subbeam.
:long_name = "Raw backscatter measurements (real part)"		
:units = "as appropriate"		Use units appropriate for the data.
angle_t echoangle_major(ping_time, beam)	MA	Electrical phase angle of the incoming echoes at time ping_time relative to the direction of each beam. Only required if the beam_type variable for this ping_time is set to split_aperture_angles.
:long_name = "Echo arrival angle in the major beam coordinate"		
:units = "arc_degree"		
float :valid_range = −180.0, 180.0		
angle_t echoangle_minor(ping_time, beam)	MA	Electrical phase angle of the incoming echoes at time ping_time relative to the direction of each beam. Only required if the beam_type variable for this ping_time is set to split_aperture_angles.
:long_name = "Echo arrival angle in the minor beam coordinate"		
:units = "arc_degree"		
float :valid_range = −180.0, 180.0		
float echoangle_major_sensitivity(beam)	MA	Scaling factor to convert electrical arrival angles into physical angles. Only required if beam_type is not set to single.
:long_name = "Major angle scaling factor"		
:units = "1"		
float :valid_min = 0.0		

Description	Obligation	Comment
float echoangle_minor_sensitivity(beam)	MA	Scaling factor to convert electrical arrival angles into physical angles. Only required if beam_type is not set to single.
:long_name = "Minor angle scaling factor"		
:units = "1"		
float :valid_min = 0.0		
float beamwidth_receive_major(ping_time, beam)	M	One-way beam width at half power down in the horizontal direction of the receive beam.
:long_name = "Half power one-way receive beam width along major (horizontal) axis of beam"		
:units = "arc_degree"		
float :valid_range = 0.0, 360.0		
int :substitute_value_used = 0		If non-zero, indicates that the variable value is a nominal value used when a measured or calibrated value is not available for a mandatory variable.
float beamwidth_receive_minor(ping_time, beam)	M	One-way beam width at half power down in the vertical direction of the receive beam.
:long_name = "Half power one-way receive beam width along minor (vertical) axis of beam"		
:units = "arc_degree"		
float :valid_range = 0.0, 360.0		
int :substitute_value_used = 0		If non-zero, indicates that the variable value is a nominal value used when a measured or calibrated value is not available for a mandatory variable.
float beamwidth_transmit_major(ping_time, tx_beam)	MA	One-way beam width at half power down in the horizontal direction of the transmit beam.
:long_name = "Half power one-way transmit beam width along major (horizontal) axis of beam"		
:units = "arc_degree"		
float :valid_range = 0.0, 360.0		
float beamwidth_transmit_minor(ping_time, tx_beam)	MA	One-way beam width at half power down in the vertical direction of the transmit beam.
:long_name = "Half power one-way transmit beam width along minor (vertical) axis of beam"		
:units = "arc_degree"		
float :valid_range = 0.0, 360.0		
float rx_beam_rotation_phi(ping_time, beam)	M	The intrinsic *z*–*y*'–*x*" clockwise rotation about the *x*-axis of the platform coordinate system needed to give the receive beam coordinate system. For ships and similar, if installation angles are close to zero, this rotation usually matches the beam pointing angle in the across track direction.
:long_name = "receive beam angular rotation about the *x* axis"		

Description	Obligation	Comment
:units = "arc_degree"		
float :valid_range = −180.0, 180.0		
float rx_beam_rotation_theta(ping_time, beam)	M	The intrinsic *z*–*y*'–*x*" clockwise rotation about the *y*-axis of the platform coordinate system needed to give the receive beam coordinate system. For ships and similar, if installation angles are close to zero, this rotation usually matches the beam pointing angle in the along track direction.
:long_name = "receive beam angular rotation about the *y* axis"		
:units = "arc_degree"		
float :valid_range = −90.0, 90.0		
float rx_beam_rotation_psi(ping_time, beam)	M	The intrinsic *z*–*y*'–*x*" clockwise about the *z*-axis of the platform coordinate system needed to give the receive beam coordinate system. For most cases this angle is set to zero.
:long_name = "receive beam angular rotation about the *z* axis"		
:units = "arc_degree"		
float :valid_range = −180.0, 180.0		
float tx_beam_rotation_phi(ping_time, tx_beam)	M	The intrinsic *z*–*y*'–*x*" clockwise rotation about the *x*-axis of the platform coordinate system needed to give the transmit beam coordinate system. For ships and similar, if installation angles are close to zero, this rotation usually matches the beam pointing angle in the across track direction.
:long_name = "transmit beam angular rotation about the *x* axis"		
:units = "arc_degree"		
float :valid_range = −180.0, 180.0		
float tx_beam_rotation_theta(ping_time, tx_beam)	M	The intrinsic *z*–*y*'–*x*" clockwise about the *y*-axis of the platform coordinate system needed to give the transmit beam coordinate system. For ships and similar, if installation angles are close to zero, this rotation usually matches the beam pointing angle in the along track direction.
:long_name = "transmit beam angular rotation about the *y* axis"		
:units = "arc_degree"		
float :valid_range = −90.0, 90.0		
float tx_beam_rotation_psi(ping_time, tx_beam)	M	The intrinsic *z*–*y*'–*x*" clockwise about the *z*-axis of the platform coordinate system needed to give the transmit beam coordinate system. For most cases this angle is set to zero.
:long_name = "transmit beam angular rotation about the *z* axis"		
:units = "arc_degree"		
float :valid_range = −180.0, 180.0		
beam_stabilisation_t beam_stabilisation(ping_time)	M	Indicates whether or not sonar beams have been compensated for platform motion.

Description	Obligation	Comment
:long_name = "Beam stabilisation applied (or not)"		
:coordinates = "ping_time platform_latitude platform_longitude"		
beam_t beam_type	M	Type of split-aperture beam (or not).
:long_name = "Type of beam"		
float equivalent_beam_angle(ping_time, beam)	M	Equivalent beam angle.
:long_name = "Equivalent beam angle"		
:units = "sr"		
float :valid_range = 0.0, 12.56637061435917295385		Maximum value is equivalent to 4π.
int :substitute_value_used = 0		If non-zero, indicates that the variable value is a nominal value used when a measured or calibrated value is not available for a mandatory variable.
float gain_correction(ping_time, beam)	MA	Gain correction. This parameter is set from a calibration exercise. Necessary for type 2 conversion equation.
:long_name = "Gain correction"		
:units = "dB"		
int :substitute_value_used = 0		If non-zero, indicates that the variable value is a nominal value used when a measured or calibrated value is not available for a mandatory variable.
short non_quantitative_processing(ping_time)	M	Settings of any processing that is applied prior to recording backscatter data that may prevent the calculation of calibrated backscatter. A value of 0 always indicates no such processing.
:flag_meanings		Space-separated list of non-quantitative processing setting words or phrases. The first item must always be the no non-quantitative processing setting and subsequent items as appropriate to the sonar and data(e.g. ”no_non_quantitative_processing simrad_noise_filter_weak simrad_noise_filter_medium simrad_noise_filter_strong”).
short :flag_values		List of unique values (e.g. 0, 1, 3, 4) that indicate different non-quantitative processing settings that could be present in the sonar data. Must have the same number of values as settings given in the flag_meanings attribute.
:long_name = "Presence or not of non-quantitative processing applied to the backscattering data (sonar specific)"		
:coordinates = "ping_time platform_latitude platform_longitude"		
float receiver_sensitivity(ping_time, beam)	MA	Sensitivity of the sonar receiver for the current ping. Necessary for type 2 conversion equation.
:long_name = "Receiver sensitivity (re 1/µPa)"		
:units = "dB"		
int :substitute_value_used = 0		If non-zero, indicates that the variable value is a nominal value used when a measured or calibrated value is not available for a mandatory variable.
float sample_interval(ping_time)	M	Time between individual samples along a beam. Common for all beams in a ping.
:long_name = "Interval between recorded raw data samples"		

Description	Obligation	Comment
:units = "s"		
float :valid_min = 0.0		
:coordinates = "ping_time platform_latitude platform_longitude"		
float sample_time_offset(ping_time, tx_beam)	M	Time offset applied to sample time-stamps and intended for applying a range correction (e.g. as caused by signal processing delays). Positive values reduce the calculated range to a sample. The range of a given sample at index sample_index and if a constant sound speed is applied is given by range= sound_speed_at_transducer*(blanking_interval+sample_index*sample_interval - sample_time_offset)/2
:long_name = "Time offset that is subtracted from the timestamp of each sample"		
:units = "s"		
float blanking_interval(ping_time, beam)	M	Amount of time during reception where samples are discarded. The number of discarded sample is given by blanking_interval*sample_interval.
:long_name = "Amount of time during reception where samples are discarded"		
:units = "s"		
:valid_min = "0.0"		
float detected_bottom_range(ping_time, beam)	O	Range from the transducer face where the bottom detection criteria were encountered for the amplitude or the phase of the backscattered echoes. The range of the bottom at index bottom_index with a monostatic transducer and if a constant sound speed is applied is given by detected_bottom_range= sound_speed_at_transducer*(blanking_interval+bottom_index*sample_interval - sample_time_offset)/2.
:long_name = "Detected range of the bottom"		
:units = "m"		
:valid_min = "0.0"		
float transducer_impedance(ping_time, subbeam)	MA	Impedance of the transducer. This is the impedance of the load over which the transceiver measures voltage on the transducer subbeam. Necessary for conversion equation type 4.
:long_name = "Impedance of transducer"		
:units = "ohm"		
float transceiver_impedance(ping_time, subbeam)	MA	Impedance of the transceiver. This is the impedance of the transducer subbeam. Necessary for conversion equation type 4.
:long_name = "Impedance of transceiver"		
:units = "ohm"		
sample_t time_varied_gain(ping_time)	MA	Time-varied gain (TVG) used for each ping. Should contain TVG coefficient vectors. Necessary for type 2 conversion equations.
:long_name = "Time-varied-gain coefficients"		
:units = "dB"		

Description	Obligation	Comment
:coordinates = "ping_time platform_latitude platform_longitude"		
float transducer_gain(ping_time, beam, frequency)	MA	Gain of the transducer beam. This is the parameter that is set from a calibration exercise. Necessary for conversion equation type 1, 3 and 4.
:long_name = "Gain of transducer"		
:units = "dB"		
int :substitute_value_used = 0		If non-zero, indicates that the variable value is a nominal value used when a measured or calibrated value is not available for a mandatory variable.
float calibrated_frequency(frequency)	M	Frequencies for which transducer gain values have been estimated during the calibration exercise.
:long_name = "Calibration gain frequencies"		
:units = "Hz"		
float :valid_min = 0.0		
float transmit_bandwidth(ping_time, tx_beam)	O	Estimated bandwidth of the transmitted pulse. For CW pulses, this is a function of the pulse duration and frequency. For FM pulses, this will be close to the difference between transmit_frequency_start and transmit_frequency_stop.
:long_name = "Nominal bandwidth of transmitted pulse"		
:units = "Hz"		
float :valid_min = 0.0		
float transmit_duration_nominal(ping_time, tx_beam)	M	Nominal duration of the transmit pulse. This is not the effective pulse duration.
:long_name = "Nominal duration of transmitted pulse"		
:units = "s"		
float :valid_min = 0.0		
int :substitute_value_used = 0		If non-zero, indicates that the variable value is a nominal value used when a measured or calibrated value is not available for a mandatory variable.
pulse_t transmit_pulse_model_r(ping_time, tx_beam)	MA	Real part of the model of the transmit pulse. The exact shape of the theoretical transmit pulse is given at the same sampling rate of the backscatter measurements. The shape reflects both the weighting of the pulse and the filters that have been applied. The pulse shape is used for matched filtering of complex samples in type 4 conversion equations.
:long_name = "Real part of the model of the transmit pulse"		
float :valid_min = −1.0		
float :valid_max = 1.0		
pulse_t transmit_pulse_model_i(ping_time, tx_beam)	MA	Imaginary part of the model of the transmit pulse. The exact shape of the theoretical transmit pulse is given at the same sampling rate as the backscatter measurements. The shape reflects both the weighting of the pulse and the filters that have been applied. The pulse shape is used for matched filtering of complex samples in type 4 conversion equations.

Description	Obligation	Comment
:long_name = "Imaginary part of the model of the transmit pulse"		
float :valid_min = −1.0		
float :valid_max = 1.0		
float receive_duration_effective(ping_time, tx_beam)	MA	Effective duration of the received pulse. This is the duration of the square pulse containing the same energy as the actual receive pulse. This parameter is either theoretical or comes from a calibration exercise and adjusts the nominal duration of the transmitted pulse to the measured one. During calibration it is obtained by integrating the energy of the received signal on the calibration target normalised by its maximum energy. Necessary for type 1, 2, 3 and 4 conversion equations.
:long_name = "Effective duration of received pulse"		
:units = "s"		
float :valid_min = 0.0		
int :substitute_value_used = 0		If non-zero, indicates that the variable value is a nominal value used when a measured or calibrated value is not available for a mandatory variable.
float transmit_frequency_start(ping_time, tx_beam)	M	Frequency at the start of the transmit pulse. The beam dimension can be omitted, in which case the value apples to all beams in the ping.
:long_name = "Start frequency in transmitted pulse"		
:standard_name = "sound_frequency"		
:units = "Hz"		
float :valid_min = 0.0		
float transmit_frequency_stop(ping_time, tx_beam)	M	Frequency at the end of the transmit pulse. The beam dimension can be omitted, in which case the value apples to all beams in the ping.
:long_name = "Stop frequency in transmitted pulse"		
:standard_name = "sound_frequency"		
:units = "Hz"		
float :valid_min = 0.0		
float transmit_power(ping_time, tx_beam)	MA	Electrical transmit power used for the ping. Necessary for type 1 conversion equations
:long_name = "Nominal transmit power"		
:units = "W"		
float :valid_min = 0.0		
float transmit_source_level(ping_time, tx_beam)	MA	Source level generated by the transmit ping. Necessary for type 2 conversion equations.
:long_name = "Transmit source level (re 1 µPa at 1 m)"		

Description	Obligation	Comment
:units = "dB"		
int :substitute_value_used = 0		If non-zero, indicates that the variable value is a nominal value used when a measured or calibrated value is not available for a mandatory variable.
float transmitter_and_receiver_coefficient(ping_time)	MA	Sum of transmit source level (dB re 1 µPa at 1 m), voltage sensitivity (dB re 1 V/µPa), and system gain (dB). Necessary for type 6 conversion equations.
:long_name = "Transmitter and receiver coefficient"		
:units = "dB"		
int :substitute_value_used = 0		If non-zero, indicates that the variable value is a nominal value used when a measured or calibrated value is not available for a mandatory variable.
transmit_t transmit_type(ping_time, tx_beam)	M	Type of transmit pulse.
:long_name = "Type of transmitted pulse"		
int receive_transducer_index(beam)	MA	Receiving or monostatic transducer index associated with the given beam
:valid_min = "0"		
:long_name = "Receive transducer index"		
int transmit_transducer_index(ping_time, tx_beam)	MA	Transmitting or monostatic transducer index associated with the given transmit beam
:valid_min = "0"		
:long_name = "Transmit transducer index"		
int transmit_beam_index(ping_time, beam)	MA	Transmit beam index associated with the given beam
:valid_min = "0"		
:long_name = "Transmit beam index associated with the given beam"		
int active_MRU(ping_time)	MA	Indicate the index of the MRU sensor used at the time of the ping to compute the platform attitude.
:valid_min = "0"		
:long_name = "Active MRU sensor index"		
:coordinates = "ping_time platform_latitude platform_longitude"		
int active_position_sensor(ping_time)	MA	Indicate the index of the position sensor used at the time of the ping to compute the platform position.
:valid_min = "0"		
:long_name = "Active position sensor index"		
:coordinates = "ping_time platform_latitude platform_longitude"		

Description	Obligation	Comment
float sound_speed_at_transducer(ping_time)	O	Sound speed at transducer depth at the time of the ping
:long_name = "Indicative sound speed at ping time and transducer depth"		
:units = "m/s"		
float :valid_min = 0.0		
:standard_name = "speed_of_sound_in_sea_water"		
:coordinates = "ping_time platform_latitude platform_longitude"		
double platform_latitude(ping_time)	M	Latitude of the platform reference point in WGS-84 reference system at the time of the ping.
double :valid_range = −90.0, 90.0		
:standard_name = "Platform latitude"		
:units = "degrees_north"		
:long_name = "latitude"		
:coordinates = "ping_time platform_latitude platform_longitude"		
double platform_longitude(ping_time)	M	Longitude of the platform reference point in WGS-84 reference system at the time of the ping.
double :valid_range = −180.0, 180.0		
:standard_name = "Platform longitude"		
:units = "degrees_east"		
:long_name = "longitude"		
:coordinates = "ping_time platform_latitude platform_longitude"		
float platform_heading(ping_time)	M	Heading of the platform at time of the ping.
:standard_name = "platform_orientation"		
:units = "degrees_north"		
:long_name = "Platform heading(true)"		
float :valid_range = 0, 360.0		
:coordinates = "ping_time platform_latitude platform_longitude"		
float platform_pitch(ping_time)	M	Platform pitch at the time of the ping.
:standard_name = "platform_pitch_angle"		
:units = "arc_degree"		
:long_name = "pitch angle"		
float :valid_range = −90.0, 90.0		

Description	Obligation	Comment
:coordinates = "ping_time platform_latitude platform_longitude"		
float platform_roll(ping_time)	M	Platform roll at the time of the ping.
:standard_name = "platform_roll_angle"		
:units = "arc_degree"		
:long_name = "roll angle"		
:coordinates = "ping_time platform_latitude platform_longitude"		
float platform_vertical_offset(ping_time)	M	Distance from the platform reference point to the water line (distance are positives downwards). For ships and similar, this is called heave and is added to the dynamic draught at the time of the ping but the concept applies equally well to underwater vehicle depth.
:long_name = "Platform vertical distance from reference point to the water line"		
:units = "m"		
:coordinates = "ping_time platform_latitude platform_longitude"		
float tx_transducer_depth(ping_time)	O	Tx transducer depth below waterline at time of the ping (distance are positives downwards). This variable can be recomputed in most cases by applying lever arm and rotation matrix taking into account for roll and pitch, platform_vertical_offset but can also take into account for drop keel position
:long_name = "Tx transducer depth below waterline"		
:units = "m"		
:coordinates = "ping_time platform_latitude platform_longitude"		
float waterline_to_chart_datum(ping_time)	O	Vertical translation vector at the time of the ping matching the distance from the water line to the chart data reference (typically Lowest Astronomical Tide or Mean Sea Level). This variable is the vector that contains the tide and allows for the positioning of samples in an absolute reference system.
:long_name = "vertical translation from waterline to chart datum reference "		
:units = "m"		
:coordinates = "ping_time platform_latitude platform_longitude"		
:vertical_coordinate_reference_system = "MSL depth"		The vertical datum to which distance are referred to. Possible values are 'MSL Depth' or 'LAT Depth'
Subgroups		
ADCP	O	Subgroup containing ADCP calculated current velocity data.
SingleTarget	O	Subgroup containing split-beam detected single-target data.

3.9.6. Proposed Single Target additions

This group contains the single target detections from split-aperture beams positioned with their range and split-aperture arrival angles. Target Strength (TS) is calculated using conversion equations detailed in Chapter 4. Algorithm for single target detection and parameters are detailed in Soule *et al.* (1997). The netCDF4 group name is /Sonar/Beam_group/SingleTarget.

Table 12. Description of the single_target subgroup of the beam group.

Description	Obligation	Comment
Types		
int(*) target_identifier		Variable length vector used to store array of index of single target data detections.
float(*) target_range		Variable length vector used to store array of range of single target data detections.
float(*) target_angle		Variable length vector used to store variable length array of angle of single target detections.
float(*) target_TS		Variable length vector used to store variable length array of Target Strength of single target detections.
Variables		
int single_target_count(ping_time, beam)	O	The number of single target detection in each beam/ping in the single_target_XXX variables. This value is not essential, but software that reads the vlen variables can use it to significantly improve data loading times.
:long_name = "Number of single target detection in each beam, per ping"		
int :valid_min = 0		
target_identifier single_target_identifier(ping_time,beam)	MA	Label of single target detected and possibly tracked on multiple pings. All single targets have a new unique identifier when detected, if a tracking algorithm is used the identifier of a new target can be changed to an existing one so that it can be identified as the same target.
:long_name = "Index of single target detected"		
target_range single_target_range(ping_time,beam)	MA	Range from the transducer face where the single target detection criteria were encountered for the amplitude and the phase of the backscattered echoes. Calculation of the range is similar to detected_bottom_range defined in the BeamGroup with possible weighting of the range by the backscattervalue within the pulse shape.
:units = "m"		
:long_name = "Range of single target detected"		
target_angle single_target_alongship_angle(ping_time,beam)	MA	Angle of single target detected in the minor beam coordinate.
:units = "arc_degree"		
:long_name = "Single target arrival angle in the minor beam coordinate"		
float :valid_range = −180.0, 180.0		

Description	Obligation	Comment
target_angle single_target_athwartship_angle(ping_time,beam)	MA	Angle of single target detected in the major beam coordinate.
:units = "arc_degree"		
:long_name = "Single target arrival angle in the major beam coordinate"		
float :valid_range = −180.0, 180.0		
target_TS compensated_TS(ping_time,beam,frequency)	MA	Calculated beam compensated Target Strength.
:units = "dB"		
:long_name = "Calculated Target Strength (re 1 m^2) after compensation for off-axis angle for each frequency of the receive echo from spectral analysis of the FM pulse or frequency of the CW pulse"		
target_TS uncompensated_TS(ping_time,beam,frequency)	MA	Calculated beam uncompensated Target Strength.
:units = "dB"		
:long_name = "Calculated Target Strength (re 1 m^2) uncompensated for off-axis angle for each frequency of the receive echo from spectral analysis of the FM pulse or frequency of the CW pulse"		
float param_TS_threshold(ping_time,beam)	MA	TS threshold for single target detection
:units = "dB"		
:long_name = "Minimum TS (re 1 m^2) threshold for single target detection"		
float param_gain_compensation(ping_time,beam)	MA	Gain compensation for single target detection
:units = "dB"		
:long_name = "Maximum one-way angular gain compensation for single target detection"		
float param_minimum_echo_duration(ping_time,beam)	MA	Minimimum echo duration for single target detection
:long_name = "Minimimum normalized echo duration for single target detection relative to nominal pulse duration"		
float param_maximum_echo_duration(ping_time,beam)	MA	Maximimum echo duration for single target detection
:long_name = "Maximimum normalized echo duration for single target detection relative to nominal pulse duration"		

Description	Obligation	Comment
float param_maximum_phase_deviation(ping_time, beam)	MA	Maximimum phase deviation for single target detection
:units = "arc_degree"		
:long_name = "Maximimum phase standard deviation for single target detection"		
float param_minimum_echo_spacing(ping_time,beam)	MA	Minimum echo spacing for single target detection
:long_name = "Minimum distance between two single targets detected relative to nominal pulse duration"		
float param_TSf_processing_window_duration(ping_time, beam)	MA	TS(f) processing window duration.
:units = "s"		
:long_name = "Duration of the processing window for spectral analysis around the peak echo value for single target detection"		

3.9.7. Proposed ADCP additions

This group contains the ADCP beam data, single ping calculations and associated metadata. The netCDF4 group name is /Sonar/Beam_groupX/ADCP.

Table 13. Description of the ADCP subgroup of the BeamGroup1 group.

Description	Obligation	Comment
Types		
float(*) sample_v		Variable length vector used to store ragged arrays of velocity data. Type added for ADCP.
Variables		
float backscatter_at_bottom_i(ping_time, beam)	MA	Backscatter value(imaginary part) at detected bottom, in each beam.
:units = "W"		
:long_name = "Raw backscatter at bottom (imaginary part)"		
float backscatter_at_bottom_r(ping_time, beam)	MA	Backscatter value(real part) at detected bottom, in each beam.
:units = "W"		
:long_name = "Raw backscatter at bottom (real part)"		
float bin_lenght(ping_time)	MA	Distance covered by transmit pulse.
:units = "m"		
:long_name = "Distance covered by transmit pulse"		
float :valid_range = 0.0, 20.0		
float bottom_track_velocity_vessel_x(ping_time)	MA	Calculated bottom track velocity, pos in vessel coordinate x direction, forward.
:units = "m/s"		
:long_name = "Calculated bottom track velocity value relative own vessel, direction x"		
float :valid_range = -50.0, 50.0		
float bottom_track_velocity_vessel_y(ping_time)	MA	Calculated bottom track velocity, pos in vessel coordinate y direction, starboard.
:units = "m/s"		
:long_name = "Calculated bottom track velocity value relative own vessel, direction y"		
float :valid_range = -50.0, 50.0		

Description	Obligation	Comment
float bottom_track_velocity_vessel_z(ping_time)	MA	Calculated bottom track velocity, pos in vessel coordinate z direction, down.
:units = "m/s"		
:long_name = "Calculated bottom track velocity value relative own vessel, direction z"		
float :valid_range = -50.0, 50.0		
sample_t correlation(ping_time, beam)	MA	Calculated beam correlation. Each element in the 2D matrix is a variable length vector (of type sample_t)that contains the calculated correlation along that beam and ping time
:units = "%"		
:long_name = "Calculated beam correlation"		
float :valid_range = 0.0, 100.0		
int correlation_at_bottom(ping_time, beam)	MA	Correlation at detected bottom, in each beam.
:units = "%"		
:long_name = "Calculated beam correlation at bottom depth"		
int :valid_range = 0, 100		
float correlation_factor_limit(ping_time)	MA	Filtering parameter. Element discarded if correlation below given limit.
:units = "%"		
:long_name = "Correlation minimum limit used for filtering current ping"		
float :valid_range = 0.0, 100.0		
sample_v current_velocity_geographical_down(ping_time)	MA	Calculated water column current velocity, pos in down direction.
:units = "m/s"		
:long_name = "Calculated water current velocity values for the geographical down direction"		
float :valid_range = -50.0, 50.0		
sample_v current_velocity_geographical_east(ping_time)	MA	Calculated water column current velocity, pos in east direction.
:units = "m/s"		
:long_name = "Calculated water current velocity values for the geographical east direction"		
float :valid_range = -50.0, 50.0		

Description	Obligation	Comment
sample_v current_velocity_geographical_north(ping_time)	MA	Calculated water column current velocity, pos in north direction.
:units = "m/s"		
:long_name = "Calculated water current velocity values for the geographical north direction"		
float :valid_range = -50.0, 50.0		
sample_v current_velocity_vessel_x(ping_time)	O	Calculated vessel relative water column current velocity, in x direction.
:units = "m/s"		
:long_name = "Calculated water current velocity values relative own vessel, direction x"		
float :valid_range = -50.0, 50.0		
sample_v current_velocity_vessel_y(ping_time)	O	Calculated vessel relative water column current velocity, in y direction.
:units = "m/s"		
:long_name = "Calculated water current velocity values relative own vessel, direction y"		
float :valid_range = -50.0, 50.0		
sample_v current_velocity_vessel_z(ping_time)	O	Calculated vessel relative water column current velocity, in z direction.
:units = "m/s"		
:long_name = "Calculated water current velocity values relative own vessel, direction z"		
float :valid_range = -50.0, 50.0		
float depth_first_sample_center(ping_time)	MA	Depth from surface to first valid sample center.
:units = "m"		
:long_name = "Depth from surface to first valid sample center"		
sample_v error_velocity(ping_time)	O	Calculated error velocity, Quality factor, low value indicates homogenous water current layer and high measurement quality.
:units = "m/s"		
:long_name = "Calculated error velocity"		
float :valid_min = 0.0		
float error_velocity_limit(ping_time)	MA	Filtering parameter. Vessel and geographical current velocity sample discarded if error velocity is above given limit.
:units = "m/s"		
:long_name = "Error velocity maximum limit used for filtering current ping"		

Description	Obligation	Comment
float :valid_range = 0.0, 100.0		
float transmit_duration_nominal_sub_pulse(ping_time)	MA	Nominal duration of the transmit sub-pulse. This is in the case where the transmitted pulse consists of a series of sub-pulses. ADCP specific.
:units = "s"		
:long_name = "Nominal duration of the transmitted sub-pulse"		
float :valid_min = 0.0		
float transmit_lag_interval_sub_pulse(ping_time)	MA	Lag interval between the transmitted sub-pulse.This is in the case where the transmitted pulse consists of a series of sub-pulses. ADCP specific.
:units = "s"		
:long_name = "Lag interval of the transmitted sub-pulse"		
float :valid_min = 0.0		
int quality(ping_time)	MA	Quality percent for each depth cell.
:units = "%"		
:long_name = "Quality indicator for the water current velocity calculation"		
int :valid_range = 0, 100		
float scaling_factor	MA	Scaling factor from ADCP calibration used in velocity calculations.
:long_name = "Scaling factor for velocity calculations"		
float :valid_range = 0.0, 2.0		
float slant_range_to_bottom(ping_time, beam)	MA	Detected bottom in each beam.
:units = "m"		
:long_name = "Slant range to bottom for each beam"		
int sv_dbw_high_limit(ping_time)	MA	Filtering parameter. Element discarded if backscatter is above given limit.
:units = "dB"		
:long_name = "Sv (re 1 m^{-1}) maximum limit used for filtering current ping"		
int :valid_range = -235, 0		
int sv_dbw_low_limit(ping_time)	MA	Filtering parameter. Element discarded if backscatter is below given limit.
:units = "dB"		
:long_name = "Sv (re 1 m^{-1}) minimum limit used for filtering current ping"		

Description	Obligation	Comment
int :valid_range = -235, 0		
sample_v velocity(ping_time, beam)	O	Calculated beam velocity. Each element in the 2D matrix is a variable length vector that contains the calculated velocity along that beam and ping time.
:units = "m/s"		
:long_name = "Calculated beam velocity"		
float :valid_range = -50.0, 50.0		
beam_stabilisation_t velocity_depth_stabilisation(ping_time)	MA	Indicates whether samples used for velocity vector calculation have been compensated for platform motion.
:long_name = "Velocity depth stabilization applied (or not) "		
beam_stabilisation_t velocity_motion_stabilisation(ping_time)	MA	Indicates whether beam velocities have been compensated for platform motion.
:long_name = "Velocity motion stabilization applied (or not) "		
float vertical_sample_interval(ping_time)	M	True vertical distance between calculated geographical or vessel relative current values.
:units = "m"		
:long_name = "Distance between recorded data samples"		
float :valid_min = 0.0		
Subgroups		
Mean_current	O	Subgroup containing calculated averaged current velocity data.

This group contains the calculated averaged current velocity data and associated metadata. The netCDF4 group name is Mean_current and is a subgroup of ADCP group.

Table 14. Description of the Mean current subgroup of the ADCP group.

Description	Obligation	Comment
Types		
uint64(*) ping_t		Variable length vector used to store ping time of pings used in averaging.
Dimensions		
mean_time = unlimited		Can be of fixed or unlimited length, as appropriate.
Coordinate variables		
uint64 mean_time(mean_time)	M	Timestamp for the center of an averaging period.

Description	Obligation	Comment
:axis = "T"		
:calendar = "gregorian"		
:long_name = "Time-stamp of each ping"		
:standard_name = "time"		
:units = "nanoseconds since 1970-01-01 00:00:00Z" or "nanoseconds since 1601-01-01 00:00:00Z"		
Variables		
int averaging(mean_time)	M	Number of ping averages.
:long_name = "Number of ping averages"		
int :valid_min = 1		
float bottom_track_velocity_vessel_x(mean_time)	MA	Calculated bottom track velocity, pos in vessel coordinate x direction, forward.
:units = "m/s"		
:long_name = "Calculated mean bottom track velocity value relative own vessel, direction x"		
float :valid_range = -50.0, 50.0		
float bottom_track_velocity_vessel_y(mean_time)	MA	Calculated bottom track velocity, pos in vessel coordinate y direction, starboard.
:units = "m/s"		
:long_name = "Calculated mean bottom track velocity value relative own vessel, direction y"		
float :valid_range = -50.0, 50.0		
float bottom_track_velocity_vessel_z(mean_time)	MA	Calculated bottom track velocity, pos in vessel coordinate z direction, down.
:units = "m/s"		
:long_name = "Calculated mean bottom track velocity value relative own vessel, direction z"		
float :valid_range = -50.0, 50.0		
sample_v current_velocity_geographical_down(mean_time)	MA	Calculated water column current velocity, pos in down direction.
:units = "m/s"		
:long_name = "Calculated mean water current velocity values for the geographical down direction"		
float :valid_range = -50.0, 50.0		

Description	Obligation	Comment
sample_v current_velocity_geographical_east(mean_time)	MA	Calculated water column current velocity, pos in east direction.
:units = "m/s"		
:long_name = "Calculated mean water current velocity values for the geographical east direction"		
float :valid_range = -50.0, 50.0		
sample_v current_velocity_geographical_north(mean_time)	MA	Calculated water column current velocity, pos in north direction.
:units = "m/s"		
:long_name = "Calculated mean water current velocity values for the geographical north direction"		
float :valid_range = -50.0, 50.0		
sample_v current_velocity_vessel_x(mean_time)	MA	Calculated vessel relative water column current velocity, in x direction.
:units = "m/s"		
:long_name = "Calculated mean water current velocity values relative own vessel, direction x"		
float :valid_range = -50.0, 50.0		
sample_v current_velocity_vessel_y(mean_time)	MA	Calculated vessel relative water column current velocity, in y direction.
:units = "m/s"		
:long_name = "Calculated mean water current velocity values relative own vessel, direction y"		
float :valid_range = -50.0, 50.0		
sample_v current_velocity_vessel_z(mean_time)	MA	Calculated vessel relative water column current velocity, in z direction.
:units = "m/s"		
:long_name = "Calculated mean water current velocity values relative own vessel, direction z"		
float :valid_range = -50.0, 50.0		
float mean_bin_lenght(mean_time)	MA	Distance covered by transmit pulse.
:units = "m"		
:long_name = "Distance covered by transmit pulse"		
float :valid_range = 0.0, 20.0		

Description	Obligation	Comment
double mean_platform_heading(mean_time)	MA	Calculated mean heading for the averaging period.
:units = "degrees_north"		
:long_name = "Calculated mean heading for the averaging period."		
double :valid_range = 0.0, 360.0		
double mean_platform_latitude(mean_time)	MA	Latitude representing the averaging period.
:units = "degrees_north"		
:long_name = "Latitude representing the averaging period."		
double :valid_range = -90.0, 90.0		
double mean_platform_longitude(mean_time)	MA	Longitude representing the averaging period.
:units = "degrees_east"		
:long_name = "Longitude representing the averaging period."		
double :valid_range = -180.0, 180.0		
double mean_platform_pitch(mean_time)	MA	Calculated mean pitch for the averaging period.
:units = "arc_degree"		
:long_name = "Calculated mean pitch for the averaging period."		
double :valid_range = -90.0, 90.0		
double mean_platform_roll(mean_time)	MA	Calculated mean roll for the averaging period.
:units = "arc_degree"		
:long_name = "Calculated mean roll for the averaging period."		
double :valid_range = -90.0, 90.0		
double mean_platform_vertical(mean_time)	MA	Calculated mean vertical offset for the averaging period.
:units = "m"		
:long_name = "Calculated mean vertical offset for the averaging period. Zero if velocity_depth_stabilisation is true."		
float percent_good_limit	MA	Filtering parameter. Mean velocity value disgarded if quality below given limit.
:units = "%"		
:long_name = "Percent good limit used to filter the water current velocities"		
float :valid_range = 0.0, 100.0		
ping_t ping_averaged(mean_time)	MA	Reference to pings (by ping_time) used for averaging to find start-end of lat, lon, vessel speed, heading etc.

Description	Obligation	Comment
:long_name = "Time reference to pings used for averaging"		
int quality(mean_time)	MA	Averaged quality in percent for each depth cell.
:units = "%"		
:long_name = "Quality indicator for the water current velocity calculation"		
int :valid_range = 0, 100		

3.9.8. Gridded group

This group contains gridded sonar backscatter data and associated metadata. The netCDF4 group name is **GriddedX**, where **X** is an integer.

Table 15. Description of the beam grid subgroups of the sonar group.

Description	Obligation	Comment
Group attributes		
:beam_mode	M	Mode of the beam in this subgroup, taken from the defined vocabulary of: “vertical” (a set of beams that form a vertical slice through the water), ”horizontal” (a set of beams that form a nominally horizontal plane through the water), and ”inspection” (a set of beams with arbitrary pointing directions).
conversion_equation_t :conversion_equation_type	M	Type of equation used to convert backscatter measurements into volume backscattering strength and target strength.
Types		
byte enum backscatter_type_t {Sv = 0, Sa = 1}		In what form is the acoustic data stored? Controlled vocabulary options include : Sv, Volume backscattering strength (dB re 1 m^{-1}) and Sa, Nautical area scattering coefficient (m^2 nmi^{-2}).
byte enum range_axis_interval_type_t {Range = 0, Depth = 1}		Range axis interval by which data have been regridded. Controlled vocabulary includes: Range (metres), Depth relative to waterline (metres).
byte enum ping_axis_interval_type_t {Time_seconds = 0, Distance_nautical_miles = 1, Distance_meters = 2, Number_of_ping = 3}		Ping axis interval by which data have been regridded. Controlled vocabulary includes: Time based intervals (Time (seconds)), Distance based intervals (Distance (nautical miles); Distance (metres)), Ping based intervals (Number of pings).
Dimensions		
beam		The number of receive beams in this grid group.
tx_beam		The number of transmit beams in this grid group.
frequency		The number of frequencies in this grid group.
ping_axis		Number of cells in ping dimension.
range_axis		Number of cells in range dimension.
Variables		
string beam(beam)	M	Beam name (or number or identification code).
:long_name = "Beam name"		
float frequency(frequency)	M	Frequency.
:long_name = "Frequency of the receive echo from spectral analysis of the FM pulse or frequency of the CW pulse."		
:units = "Hz"		
float :valid_min = 0.0		
string beam_reference(frequency)	M	Beam name used for a given frequency.
:long_name = "Reference to the beam for a given frequency"		

Description	Obligation	Comment
backscatter_type_t backscatter_type(frequency)	M	Standard definition of backscatter unit.
:long_name = "Backscatter type for gridded data"		
ping_axis_interval_type_t ping_axis_interval_type	MA	Reference for regridding data in ping axis.
:long_name = "Interval type for regridding the data in ping axis"		
float ping_axis_interval_value	M	Value for data ping axis interval according to its specified type.
:long_name = "Ping axis interval for regridding the data"		
range_axis_interval_type_t range_axis_interval_type	M	Reference for regridding the data in range or depth axis.
:long_name = "Interval type for regridding the data in range axis"		
float range_axis_interval_value	M	Value for data range axis interval according to its specified type.
:long_name = "Range axis interval for regridding the data"		
uint64 cell_ping_time(ping_axis, tx_beam)	M	Mean timestamp of the pings contributing to the cell.
:axis = "T"		
:calendar = "gregorian"		
:long_name = "Mean time-stamp of each cell"		
:standard_name = "time"		
:units = "nanoseconds since 1970-01-01 00:00:00Z" or "nanoseconds since 1601-01-01 00:00:00Z"		
:coordinates = "ping_axis platform_latitude platform_longitude"		
float integrated_backscatter(ping_axis, range_axis, frequency)	M	Integrated backscatter measurement.
:long_name = "Integrated backscatter of the raw backscatter measurements sampled in this cell for each frequency."		
:units = "as appropriate"		Use units appropriate for the data.
float beamwidth_receive_major(ping_axis, beam)	M	One-way beam width at half power down in the horizontal direction of the receive beam.
:long_name = "Half power one-way receive beam width along major (horizontal) axis of beam"		
:units = "arc_degree"		
float :valid_range = 0.0, 360.0		
float beamwidth_receive_minor(ping_axis, beam)	M	One-way beam width at half power down in the vertical direction of the receive beam.

Description	Obligation	Comment
:long_name = "Half power one-way receive beam width along minor (vertical) axis of beam"		
:units = "arc_degree"		
float :valid_range = 0.0, 360.0		
float beamwidth_transmit_major(ping_axis, tx_beam)	MA	One-way beam width at half power down in the horizontal direction of the transmit beam.
:long_name = "Half power one-way transmit beam width along major (horizontal) axis of beam"		
:units = "arc_degree"		
float :valid_range = 0.0, 360.0		
float beamwidth_transmit_minor(ping_axis, tx_beam)	MA	One-way beam width at half power down in the vertical direction of the transmit beam.
:long_name = "Half power one-way transmit beam width along minor (vertical) axis of beam"		
:units = "arc_degree"		
float :valid_range = 0.0, 360.0		
float rx_beam_rotation_phi(ping_axis, beam)	M	The intrinsic *z*–*y*'–*x*" clockwise rotation about the *x*-axis of the platform coordinate system needed to give the receive beam coordinate system. For ships and similar, if installation angles are close to zero, this rotation usually matches the beam pointing angle in the across track direction.
:long_name = "receive beam angular rotation about the *x* axis"		
:units = "arc_degree"		
float :valid_range = −180.0, 180.0		
float rx_beam_rotation_theta(ping_axis, beam)	M	The intrinsic *z*–*y*'–*x*" clockwise rotation about the *y*-axis of the platform coordinate system needed to give the receive beam coordinate system. For ships and similar, if installation angles are close to zero, this rotation usually matches the beam pointing angle in the along track direction (also called tilt angle).
:long_name = "receive beam angular rotation about the *y* axis"		
:units = "arc_degree"		
float :valid_range = −90.0, 90.0		
float rx_beam_rotation_psi(ping_axis, beam)	M	The intrinsic *z*–*y*'–*x*" clockwise about the *z*-axis of the platform coordinate system needed to give the receive beam coordinate system. For most cases this angle is set to zero.
:long_name = "receive beam angular rotation about the *z* axis"		
:units = "arc_degree"		
float :valid_range = −180.0, 180.0		

Description	Obligation	Comment
float tx_beam_rotation_phi(ping_axis, tx_beam)	M	The intrinsic *z*–*y*'–*x*" clockwise rotation about the *x*-axis of the platform coordinate system needed to give the transmit beam coordinate system. For ships and similar, if installation angles are close to zero, this rotation usually matches the beam pointing angle in the across track direction.
:long_name = "transmit beam angular rotation about the *x* axis"		
:units = "arc_degree"		
float :valid_range = −180.0, 180.0		
float tx_beam_rotation_theta(ping_axis, tx_beam)	M	The intrinsic *z*–*y*'–*x*" clockwise about the *y*-axis of the platform coordinate system needed to give the transmit beam coordinate system. For ships and similar, if installation angles are close to zero, this rotation usually matches the beam pointing angle in the along track direction (also called tilt angle).
:long_name = "transmit beam angular rotation about the *y* axis"		
:units = "arc_degree"		
float :valid_range = −90.0, 90.0		
float tx_beam_rotation_psi(ping_axis, tx_beam)	M	The intrinsic *z*–*y*'–*x*" clockwise about the *z*-axis of the platform coordinate system needed to give the transmit beam coordinate system. For most cases this angle is set to zero.
:long_name = "transmit beam angular rotation about the *z* axis"		
:units = "arc_degree"		
float :valid_range = −180.0, 180.0		
beam_stabilisation_t beam_stabilisation(ping_axis)	M	Indicates whether or not sonar beams have been compensated for platform motion.
:long_name = "Beam stabilisation applied (or not)"		
:coordinates = "ping_axis platform_latitude platform_longitude"		
beam_t beam_type(beam)	M	Type of split-aperture beam (or not).
:long_name = "Type of beam"		
float equivalent_beam_angle(ping_axis, beam)	M	Equivalent beam angle.
:long_name = "Equivalent beam angle"		
:units = "sr"		
float :valid_range = 0.0, 12.566370614359172953		Maximum value is equivalent to 4π.
float gain_correction(ping_axis, beam)	MA	Gain correction. This parameter is set from a calibration exercise. Necessary for type 2 conversion equation.
:long_name = "Gain correction"		
:units = "dB"		

Description	Obligation	Comment
short non_quantitative_processing(ping_axis)	M	Settings of any processing that is applied prior to recording backscatter data that may prevent the calculation of calibrated backscatter. A value of 0 always indicates no such processing.
:flag_meanings		Space-separated list of non-quantitative processing setting words or phrases. The first item must always be the no non-quantitative processing setting and subsequent items as appropriate to the sonar and data(e.g. ”no_non_quantitative_processing simrad_noise_filter_weak simrad_noise_filter_medium simrad_noise_filter_strong”).
short :flag_values		List of unique values (e.g. 0, 1, 3, 4) that indicate different non-quantitative processing settings that could be present in the sonar data. Must have the same number of values as settings given in the flag_meanings attribute.
:long_name = "Presence or not of non-quantitative processing applied to the backscattering data (sonar specific)"		
:coordinates = "ping_axis platform_latitude platform_longitude"		
float receiver_sensitivity(ping_axis, beam)	MA	Sensitivity of the sonar receiver for the current ping. Necessary for type 2 conversion equation.
:long_name = "Receiver sensitivity (re 1/µPa)"		
:units = "dB"		
float sample_interval(ping_axis, beam)	M	Time between individual samples along a beam. Common for all beams in a ping.
:long_name = "Interval between recorded raw data samples"		
:units = "s"		
float :valid_min = 0.0		
:coordinates = "ping_axis platform_latitude platform_longitude"		
float sample_time_offset(ping_axis, tx_beam)	M	Time offset applied to sample time-stamps and intended for applying a range correction (e.g. as caused by signal processing delays). Positive values reduce the calculated range to a sample. The range of a given sample at index sample_index and if a constant sound speed is applied is given by range= sound_speed_at_transducer*(blanking_interval+sample_index*sample_interval - sample_time_offset)/2
:long_name = "Time offset that is subtracted from the timestamp of each sample"		
:units = "s"		
float blanking_interval(ping_axis, beam)	M	Amount of time during reception where samples are discarded. The number of discarded sample is given by blanking_interval*sample_interval.
:long_name = "Amount of time during reception where samples are discarded"		
:units = "s"		
:valid_min = "0.0"		
float detected_bottom_range(ping_axis, beam)	O	Range from the transducer face where the bottom detection criteria were encountered for the amplitude or the phase of the backscattered echoes. The range of the bottom at index bottom_index with a monostatic transducer and if a constant sound speed is applied is given by detected_bottom_range= sound_speed_at_transducer*(blanking_interval+bottom_index*sample_interval - sample_time_offset)/2.
:long_name = "Detected range of the bottom"		

Description	Obligation	Comment
:units = "m"		
:valid_min = "0.0"		
sample_t time_varied_gain(ping_axis)	MA	Time-varied gain (TVG) used for each ping. Should contain TVG coefficient vectors. Necessary for type 2 conversion equations.
:long_name = "Time-varied-gain coefficients"		
:units = "dB"		
:coordinates = "ping_axis platform_latitude platform_longitude"		
float transducer_gain(ping_axis, frequency)	MA	Gain of the transducer beam. This is the parameter that is set from a calibration exercise. Necessary for conversion equation type 1.
:long_name = "Gain of transducer"		
:units = "dB"		
float transmit_bandwidth(ping_axis, tx_beam)	O	Estimated bandwidth of the transmitted pulse. For CW pulses, this is a function of the pulse duration and frequency. For FM pulses, this will be close to the difference between transmit_frequency_start and transmit_frequency_stop.
:long_name = "Nominal bandwidth of transmitted pulse"		
:units = "Hz"		
float :valid_min = 0.0		
float transmit_duration_nominal(ping_axis, tx_beam)	M	Nominal duration of the transmit pulse. This is not the effective pulse duration.
:long_name = "Nominal duration of transmitted pulse"		
:units = "s"		
float :valid_min = 0.0		
float receive_duration_effective(ping_axis, tx_beam)	MA	Effective duration of the received pulse. This is the duration of the square pulse containing the same energy as the actual receive pulse. This parameter is either theoretical or comes from a calibration exercise and adjusts the nominal duration of the transmitted pulse to the measured one. During calibration it is obtained by integrating the energy of the received signal on the calibration target normalised by its maximum energy. Necessary for type 1, 2,3 and 4 conversion equations.
:long_name = "Effective duration of received pulse"		
:units = "s"		
float :valid_min = 0.0		
float transmit_frequency_start(ping_axis, tx_beam)	M	Frequency at the start of the transmit pulse. The beam dimension must be present, even if all beams have the same start frequency.
:long_name = "Start frequency in transmitted pulse"		
:standard_name = "sound_frequency"		

Description	Obligation	Comment
:units = "Hz"		
float :valid_min = 0.0		
float transmit_frequency_stop(ping_axis, tx_beam)	M	Frequency at the end of the transmit pulse. The beam dimension must be present, even if all beams have the same stop frequency.
:long_name = "Stop frequency in transmitted pulse"		
:standard_name = "sound_frequency"		
:units = "Hz"		
float :valid_min = 0.0		
float transmit_power(ping_axis, tx_beam)	MA	Electrical transmit power used for the ping. Necessary for type 1 conversion equations
:long_name = "Nominal transmit power"		
:units = "W"		
float :valid_min = 0.0		
float transmit_source_level(ping_axis, tx_beam)	MA	Source level generated by the transmit ping. Necessary for type 2 conversion equations.
:long_name = "Transmit source level (re 1 µPa at 1m)"		
:units = "dB"		
transmit_t transmit_type(ping_axis, tx_beam)	M	Type of transmit pulse.
:long_name = "Type of transmitted pulse"		
int receive_transducer_index(beam)	MA	Receiving or monostatic transducer index associated with the given beam
:valid_min = "0"		
:long_name = "Receive transducer index"		
float sound_speed_at_transducer(ping_axis)	O	Sound speed at transducer depth at the time of the ping
:long_name = "Indicative sound speed at ping time and transducer depth"		
:units = "m/s"		
float :valid_min = 0.0		
:standard_name = "speed_of_sound_in_sea_water"		
:coordinates = "ping_axis platform_latitude platform_longitude"		
double cell_latitude(ping_axis, range_axis, beam)	M	Mean latitude of the echoes contributing to the cell in WGS-84 reference system.
double :valid_range = −90.0, 90.0		

Description	Obligation	Comment
:standard_name = "Platform latitude"		
:units = "degrees_north"		
:long_name = "latitude"		
:coordinates = "ping_axis cell_latitude cell_longitude"		
double cell_longitude(ping_axis, range_axis, beam)	M	Mean longitude of the echoes contributing to the cell in WGS-84 reference system.
double :valid_range = −180.0, 180.0		
:standard_name = "Platform longitude"		
:units = "degrees_east"		
:long_name = "longitude"		
:coordinates = "ping_axis cell_latitude cell_longitude"		
float cell_depth(range_axis, beam)	M	Depth of the cell to the water line (distance are positives downwards).
:long_name = "Cell depth below water line"		
:units = "m"		
float tx_transducer_depth(ping_axis)	O	Tx transducer depth below waterline at time of the cell (distance are positives downwards).
:long_name = "Tx transducer depth below waterline"		
:units = "m"		
:coordinates = "ping_axis platform_latitude platform_longitude"		
float waterline_to_chart_datum(ping_axis)	O	Vertical translation vector at the time of the ping matching the distance from the water line to the chart data reference (typically Lowest Astronomical Tide or Mean Sea Level). This variable is the vector that contains the tide and allows for the positioning of samples in an absolute reference system.
:long_name = "vertical translation from waterline to chart datum reference "		
:units = "m"		
:coordinates = "ping_time platform_latitude platform_longitude"		
:vertical_coordinate_reference_system = "MSL depth"		The vertical datum to which distance are referred to. Possible values are 'MSL Depth' or 'LAT Depth'

3.9.9. Interpretation

This group contains the interpretation masks. the interpretation mask is independent of the ping or grid and can be used for data in both the Gridded and Beam_group. The netCDF4 group name is **InterpretationX**, where **X** is an integer.

The interpretation masks are based on a time and range, for a given a sound speed.

An proposal for storing interpretation masks in a netCDF4 structure is being developed here, with the actual netCDF4 structure provided as an example here.

more to come

3.9.10. Vendor specific group

The vendor specific group contains information about the sonar and data specific to the sonar. Data in this group must not be necessary for normal quantitative use of the sonar data. The contents of this group are at the discretion of the sonar and software that writes the SONAR-netCDF4 file. The netCDF4 group name is **/Vendor_specific**. There is no mandatory information for the vendor specific group.

Table 16. Description of the vendor specific group.

Description	Obligation	Comment
Subgroups		
SimradWBT	O	Subgroup containing optional transducer and Simrad wideband transceiver (WBT) specific information for processing and reference.

SimradWBT group

This Subgroup contains optional transducer and Simrad WBT information for processing and for reference.

Table 17. Description of the SimradWBT group in the Vendor group.

Description	Obligation	Comment
Types		
uint64(*) ping_t		Variable length vector used to store ping time of pings using the group variables.
Dimensions		
start_time = unlimited		Can be of fixed or unlimited length, as appropriate.
Coordinate variables		
uint64 start_time(start_time)	M	Timestamp for the first ping using specific values.
:axis = "T"		
:calendar = "gregorian"		
:long_name = "Time-stamp of each ping"		
:standard_name = "time"		
:units = "nanoseconds since 1970-01-01 00:00:00Z" or "nanoseconds since 1601-01-01 00:00:00Z"		
Variables		
float filter_coefficients_i(start_time, beam, stage)	MA	Imaginary part of filter coefficients.

Description	Obligation	Comment
:long_name = "Filter coefficients (imaginary part)"		
float filter_coefficients_r(start_time, beam, stage)	MA	Real part of filter coefficients.
:long_name = "Filter coefficients (real part)"		
int filter_decimation(start_time, beam, stage)	MA	Filter decimation.
:long_name = "Filter decimation"		
int :valid_min = 1		
int filter_center_frequency(start_time, beam, stage)	O	Centre frequency of the filter.
:long_name = "Filter centre frequency"		
int :valid_min = 1		
:units = "Hz"		
int filter_bandwidth(start_time, beam, stage)	O	Bandwidth of the filter passband.
:long_name = "Filter bandwidth"		
int :valid_min = 1		
:units = "Hz"		
string filter_generator_version(start_time, beam, stage)	MA	Version of the filter generator used.
:long_name = "Filter generator version"		
ping_t ping_valid(start_time)	M	Reference to pings (by ping_time) using the group variables.
:string_name = "Time reference to pings using the variables"		
float transceiver_impedance_i(start_time, beam, frequency)	M	Transceiver impedance imaginary part.
:long_name = "Transceiver impedance imaginary part"		
:units = "ohm"		
float transceiver_impedance_r(start_time, beam, frequency)	M	Transceiver impedance real part.
:long_name = "Transceiver impedance real part"		
:units = "ohm"		
int transceiver_sample_frequency(start_time, beam)	M	Transceiver sample frequency.
:long_name = "Transceiver sample frequency"		
int :valid_min = 1		
float transducer_impedance_i(start_time, beam, frequency)	M	Transducer impedance imaginary part.
:long_name = "Transducer impedance imaginary part"		
:units = "ohm"		
float transducer_impedance_r(start_time, beam, frequency)	M	Transducer impedance real part.
:long_name = "Transducer impedance real part"		
:units = "ohm"		

Chapter 4. Conversion equations

This section provides detailed formulae on how to convert the backscatter data in the Sonar group into calibrated target strength and volume backscatter strength.

4.1. Type 1

Type 1 conversion equations are used for data recorded by the Simrad SU90, SX90, and SH90 omnisonars and are presented in detail by Macaulay *et al.* (2016).

The complex-valued backscatter data are converted into calibrated target strength via:

$$TS = 10\log_{10}(P_r) + 40\log_{10}(r) + 2\alpha r - 10\log_{10}\left(\frac{P_t\lambda^2}{16\pi^2}\right) - G - 40\log_{10}(\cos\gamma), \qquad (1)$$

where P_r is linearly proportional to the received power (square of the magnitude of the complex number given by backscatter_r and backscatter_i, W) and r is the range between the transducer and target, calculated from:

$$r = \frac{c(dt \cdot i - t_0)}{2}, \qquad (2)$$

where c is sound speed (NetCDF4 variable is sound_speed_indicative, m/s), dt is the time between recorded samples (sample_interval, s), i is the sample number (from zero to one less than the number of samples), and t_0 is a time-offset (sample_time_offset-blanking_interval, s).

The absorption coefficient of sound in water is α (absorption_indicative, dB/m), P_t is the transmit power (transmit_power, W), λ is the acoustic wavelength (derived from the average of transmit_frequency_start and transmit_frequency_stop, and sound_speed_indicative, m), G is the transducer gain (transducer_gain, dB), and γ is the beam tilt angle (derived from beam_direction_x, beam_direction_y, beam_direction_z, degrees from horizontal).

The volume backscatter strength (Sv, MacLennan *et al.* (2002)) is derived from a similar equation:

$$S_v = 10\log_{10}(P_r) + 20\log_{10}(r) + 2\alpha r - 10\log_{10}\left(\frac{P_t\lambda^2 c\psi\tau_e}{32\pi^2}\right) - G - 40\log_{10}(\cos\gamma), \qquad (3)$$

where Ψ is the equivalent beam angle (equivalent_beam_angle, sr), τ_e is the effective pulse duration (*receive_duration_effective, s), and _G* is the transducer gain (transducer_gain, dB).

4.2. Type 2

Type 2 conversion equations are intended for data recorded by Furuno omnisonars. The received real-valued backscatter data are converted into calibrated target strength via:

$$TS = 20\log_{10}\left(\frac{A}{\sqrt{2}}\right) + 40\log_{10}(r) + 2\alpha r - (SL + K + \Delta G + G_T), \qquad (4)$$

where A (backscatter_r, 1) is linearly proportional to the amplitude of the received echo , r is the

range as given by equation \eqref{eq:timeToRange}, and α is the absorption coefficient of sound in water (absorption_indicative, dB/m). SL is source level (transmit_source_level (re 1µPa at 1 m), dB) and K is receiver sensitivity (receiver_sensitivity (re 1/µPa), dB). Both parameters may depend on tilt angle of the beam for transmitting and receiving.

Gain correction, ΔG, (gain_correction, dB) is determined by the calibration. Time-varied gain, G_T (time_varied_gain, dB), is given at each sample number i.

The volume backscatter strength is derived from a similar equation:

$$S_v = 20\log_{10}\left(\frac{A}{\sqrt{2}}\right) + 20\log_{10}(r) + 2\alpha r - 10\log_{10}\left(\frac{c\tau_e}{2}\Psi\right) - (SL + K + \Delta G + G_T), \qquad (5)$$

where c is sound speed (sound_speed_indicative, m/s), τ_e is the effective pulse duration (receive_duration_effective, s), and Ψ is the equivalent beam angle (equivalent_beam_angle, sr). Considering the time-varied-gain (TVG) effect on the echo shape (Sawada and Furusawa, 1993), the replacement of r by $r - r_0$, where $r_0 = \frac{1}{4}c\tau_e$, is recommended (MacLennan, 1986; Furusawa *et al.*, 1999) for calculation of S_v.

4.3. Type 3

Type 3 conversion equations are intended for data produced by Simrad EK60, ES60, and ES70 echosounders.

The backscatter power values are stored in the *backscatter_r* variable with data type *short* (16-bit signed integers) that are converted to logarithmic received power via:

$$P_r = \frac{10\log_{10}(2)}{256} P_c, \qquad (6)$$

where P_c is the compressed power value stored in the *backscatter_r* variable and P_r the logarithmic received power that is transformed into calibrated target strength via:

$$TS = P_r + 40\log_{10}(r) + 2\alpha r - 10\log_{10}\left(\frac{P_t\lambda^2}{16\pi^2}\right) - 2G_0, \qquad (7)$$

where r is the range from the transducer to the target [m], α the acoustic absorption [dB m^{-1}], P_t the transmitted power setting (*transmit_power* variable), λ the acoustic wavelength [m], and G_0 the on-axis gain (variable *transducer_gain*). Beam pattern compensation for echoes that do not arrive on the transducer axis is done in addition to this equation (if such angles can be estimated, as can be done with split-aperture transducers). Volume backscatter strength is obtained from a similar equation:

$$S_v = P_r + 20\log_{10}(r) + 2\alpha r - 10\log_{10}\left(\frac{P_t\lambda^2 c\psi\tau_e}{32\pi^2}\right) - 2G_0, \qquad (8)$$

where c is sound speed [m s^{-1}], Ψ the equivalent beam angle (variable *equivalent_beam_angle*), and τ_e the effective pulse duration (variable *receive_duration_effective*).

The effective pulse duration is computed for EK60, ES60 and ES70 from the nominal transmit pulse

duration (variable *transmit_duration_nominal*) and the Simrad term $S_{a,\,corr}$ in dB from the calibration exercise as :

$$\tau_e = \tau * 10^{\frac{2S_{a,\,corr}}{10}}, \qquad (9)$$

The range from the transducer to the target is given by:

$$tbd. \qquad (10)$$

Table 18. List of equation symbols and the matching SONAR-netCDF4 variable

Symbol	Variable	Equation type
P_c	backscatter_r	all
τ	transmit_duration_nominal	3
α	Calculated from ...	

4.4. Type 4

Type 4 conversion equations are intended for data produced by Simrad EK80 and ES80 broadband echosounders when data is recorded with complex samples.

EK80 and ES80 can record data using EK60 Power and Angle format and Type 3 equations can then be used.

In order to take advantage of the broadband mode of the EK80 echosounder, the backscatter complex amplitude samples are stored in the *backscatter_r* and the *backscatter_i* variable with data type *float* for each sub-beam corresponding to each sub-array of the transducer (Chapter 6).

For FM transmission mode, in order to increase signal-to-noise ratio and resolution along the beam axis, pulse compression is used for each sub-beam by convolving the complex backscatter received signal with a complex conjugated and time reversed version of the theoretical transmit and filtered pulse (*transmit_pulse_model*).

Noting n, the sample index in the discrete time domain, the pulse compressed complex samples for each sub-beam denoted y_{pc} are then equal to:

$$y_{pc}(n) = \frac{y_{rx}(n) * y^*_{tx,\,th}(-n)}{\|y_{tx,\,th}\|_2^2}, \qquad (11)$$

where y_{rx} is the complex received signal stored in the *backscatter_r* and *backscatter_i* variables and $y_{tx,\,th}$ is the complex theoretical transmit and filtered signal stored in *transmit_pulse_model_r* and *transmit_pulse_model_i* variables

The transceiver measures voltage over a load, $z_{\mathrm{rx,e}}$, connected in series with the transducer impedance, $z_{\mathrm{td,e}}$. The number of sub-beams N_u are stored as the *subbeam* dimension in the backscatter variables in Table 11 and received electric power is then equal to :

$$p_{\mathrm{rx,e}}(n) = \frac{1}{N_u}\left(\frac{|\sum_{u=1}^{N_u} y(n,u)|}{2\sqrt{2}}\right)^2\left(\frac{|z_{\mathrm{rx,e}} + z_{\mathrm{td,e}}|}{|z_{\mathrm{rx,e}|}}\right)^2\frac{1}{|z_{\mathrm{td,e}}|}. \qquad (12)$$

with y taken as y_{rx} for CW transmission and as y_{pc} for FM transmission.

For CW transmission, Type 3 equations can then be applied, taking logarithmic received power P_r as:

$$P_r = 10\log_{10}(p_{\mathrm{rx,e}}) \qquad (13)$$

and the nominal transmit pulse duration τ as:

$$\tau = T_{tx,\,th}\frac{mean|y_{tx,\,th}|^2}{max|y_{tx,\,th}|^2} \qquad (14)$$

with $T_{tx,\,th}$ the duration [s] of the complex theoretical transmit and filtered signal $y_{tx,\,th}$.

For FM transmission, match filtered received electric power is transformed into volume backscatter strength using:

$$S_v(n) = 10\log_{10}(p_{\mathrm{rx,e}}(n)) + 20\log_{10}(r) + 2\alpha(f_c)r - 10\log_{10}\left(\frac{P_t\lambda^2 c\psi(\frac{f_c}{f_n})^2\tau_e}{32\pi^2}\right) - 2G_0(f_c), \qquad (15)$$

where r is the range from the transducer to the sample n [m], α the acoustic absorption at the frequency f [dB m^{-1}], f_c the central frequency of the pulse bandwidth [Hz], P_t the transmitted power setting (*transmit_power* variable) [W], λ the acoustic wavelength [m], c is sound speed [m s^{-1}], ψ the equivalent beam angle at the transducer nominal frequency f_n (variable *equivalent_beam_angle*), τ_e the effective pulse duration (variable *receive_duration_effective*) and G_0 the on-axis gain (variable *transducer_gain*) from the calibration exercise at the frequency f_c.

The match filtered effective pulse duration τ_e is computed for EK80 and ES80 from

$$\tau_e = T_{tx,\,pc}.\frac{mean|y_{tx,\,pc}|^2}{max|y_{tx,\,pc}|^2}, \qquad (16)$$

where $y_{tx,\,pc}$ is the autocorrelation of the theoretical transmit pulse

$$y_{tx,\,pc}(n) = \frac{y_{tx,\,th}(n) * y^*_{tx,\,th}(-n)}{\|y_{tx,\,th}\|_2^2}, \qquad (17)$$

and $T_{tx,\,pc}$ the $y_{tx,\,pc}$ duration [s]

The use of broadband FM tranmission enables to provide volume backscatter strength as a function of frequency, through:

$$S_v(f) = 10\log_{10}(p_{\mathrm{rx,e}}(f)) + 20\log_{10}(r) + 2\alpha(f)r - 10\log_{10}\left(\frac{y_{tx,\,pc}(f)P_t\lambda^2 c\psi(\frac{f_n}{f})^2\tau_{FFT}}{32\pi^2}\right) - 2G_0(f), \qquad (18)$$

where $p_{\mathrm{rx,e}}(f)$ and $y_{tx,\,pc}(f)$ are the Fourier transforms at frequency f of $p_{\mathrm{rx,e}}$ and $y_{tx,\,pc}$, performed with the same duration/windowing characteristics, r is the range [m] from the transducer to the

sample centered in the Fourier transform window , and τ_{FFT} the duration of applied Fourier transform.

Single target strength as a function of frequency is obtained from a similar equation, for targets located on beam-axis:

$$TS(f) = 10\log_{10}(p_{\mathrm{rx,e}}(f)) + 40\log_{10}(r) + 2\alpha(f)r - 10\log_{10}\left(\frac{y_{tx,\ pc}(f)P_t\lambda^2}{16\pi^2}\right) - 2G_0(f), \qquad (19)$$

Beam pattern compensation for echoes that do not arrive on the transducer axis may be added to this equation, if such angles are estimated, as can be done with split-aperture transducers.

4.5. Type 5

Type 5 conversion equation is intended for sonars for which the sonar manufacturer computes standard TS and Sv values as defined in MacLennan et al. (2002).

Using equation 5, the Sv value is directly stored in the *backscatter_r* variable and/or the TS value is directly stored in the *backscatter_i* variable. It is permitted to store only Sv or only TS values, meaning that a file could have just a *backscatter_r* variable, just a *backscatter_i* variable, or both.

4.6. Type 6

Type 6 conversion equations are intended for data recorded by Furuno FCV-38 echosounder.

The echosouder has a split-aperture transducer with type A arrangement (see Section 6.2.1). It outputs four complex backscatter signals (backscatter_r, backscatter_i, 1) from beam number 0, 1, 2, and 3. The signals from beam number 0 and 1 are defined by $z_0 = y_3 + y_4$ and $z_1 = y_1 + y_2$, respectively, where y_x is the complex signal from quadrant x. Similarly, the signals from beam number 2 and 3 are defined by $z_2 = y_2 + y_3$ and $z_3 = y_1 + y_4$, respectively.

Two complex signals, z_0 and z_1, are converted into the calibrated target strength via:

$$TS = 20\log_{10}\left(\frac{A}{\sqrt{2}}\right) + 40\log_{10}(r) + 2\alpha r - (TR + \Delta G), \qquad (20)$$

where A is the amplitude or envelope of the received voltage signal (V), calculated from:

$$I = \mathrm{Re}(z_0) + \mathrm{Re}(z_1), \qquad (21)$$

$$Q = \mathrm{Im}(z_0) + \mathrm{Im}(z_1), \qquad (22)$$

$$A = \frac{4\sqrt{I^2 + Q^2}}{2^{32} - 1}. \qquad (23)$$

The range between the transducer and target is r, calculated from:

$$r = \frac{c(dt \cdot i - t_0)}{2}, \qquad (24)$$

where c is the sound speed (sound_speed_indicative, m/s), dt is the time between recorded samples (sample_interval, s), i is the sample number (from zero to one less than the number of samples), and t_0 is the time-offset (sample_time_offset - blanking_interval, s).

The absorption coefficient of sound in water is α (absorption_indicative, dB/m). TR is the transmitter and receiver coefficient (transmitter_and_receiver_coefficient, dB). The gain correction, ΔG (gain_correction, dB), is determined by the calibration. Beam pattern compensation for echoes that do not arrive on the transducer axis is done in addition to equation \eqref{eq:FCV38TS}.

The volume backscatter strength is derived from a similar equation:

$$S_v = 20\log_{10}\left(\frac{A}{\sqrt{2}}\right) + 20\log_{10}(r) + 2\alpha r - 10\log_{10}\left(\frac{c\,\tau_e\,\psi}{2}\right) - (TR + \Delta G), \qquad (25)$$

where τ_e is the effective pulse duration (receive_duration_effective, s) and Ψ is the equivalent beam angle (equivalent_beam_angle, sr).

Chapter 5. Coordinate systems

This section provides a complete mathematical overview of the coordinate systems necessary to physically locate the position of an echo relative to a geographic coordinate system when measured from a moving or stationary platform. The coordinate systems detailed here follow those used in some multibeam bathymetric mapping sonars, but in many cases this level of preciseness will be unnecessary and several coordinate systems will overlap.

There are four right-handed Cartesian coordinate systems (Table 19, Figure 2) associated with the platform, transducer(s), and acoustic beams. Some of the coordinate systems have different origins and this is implemented as a translation vector in the input coordinate system. Coordinate system rotation angles are as per the right-hand rule.

There is also a geographic coordinate system (Figure 2) that provides for location of the platform on Earth and also the height above a datum (e.g. mean sea level). The platform heading variable (degrees clockwise from north) can be used to obtain the sonar orientation in the geographic coordinate system (this applies to both stationary and mobile sonar platforms).

Table 19. List of coordinate systems used to physically locate the position of received echoes.

Coordinate system name	**Origin**	***x*-axis**	***y*-axis**	***z*-axis**
Surface coordinate system (SCS)	Platform origin	Pointing horizontally towards North	Pointing horizontally towards East	Pointing down with gravity
Platform coordinate system (PCS)	Platform origin	Parallel to the main axis of the platform, positive values toward the front of the platform	Perpendicular to the main axis of the platform, positive values to the starboard side	Pointing down parallel to platform mast
Transducer coordinate system (TCS)	Centre of transducer face			
not stabilized		Pointing forward along transducer plane	Pointing starboard along transducer plane	Pointing down orthogonal to transducer plane
stabilized		Pointing forward horizontally	Pointing horizontally and perpendicular to x-axis	Pointing down with gravity
Sonar Beam coordinate system (BCS)	Centre of transducer face	Arbitrary	Orthogonal to the *x*-axis	Pointing along the beam axis

The platform coordinate system (PCS) is obtained from the transducer coordinate system (TCS) through a translation of the PCS origin and a rotation to align with the transducer face. Transducer depth and lever arms are applied in this coordinate system.

The surface coordinate system (SCS) is obtained from the platform coordinate system (PCS) through a rotation that corresponds to the yaw/heading, pitch and roll of the platform. No translation is necessary. Platform heave is not applied in this coordinate system.

NOTE If a sonar actively alters beam pointing directions to compensate for motion of the platform including static installation offsets, the *beam_stabilisation* variable is set to True, otherwise to False. If True, the transducer coordinate system (TCS) is obtained from the SCS only through a translation of SCS origin, the transducer adapts its beam in order to compensate for ship movements

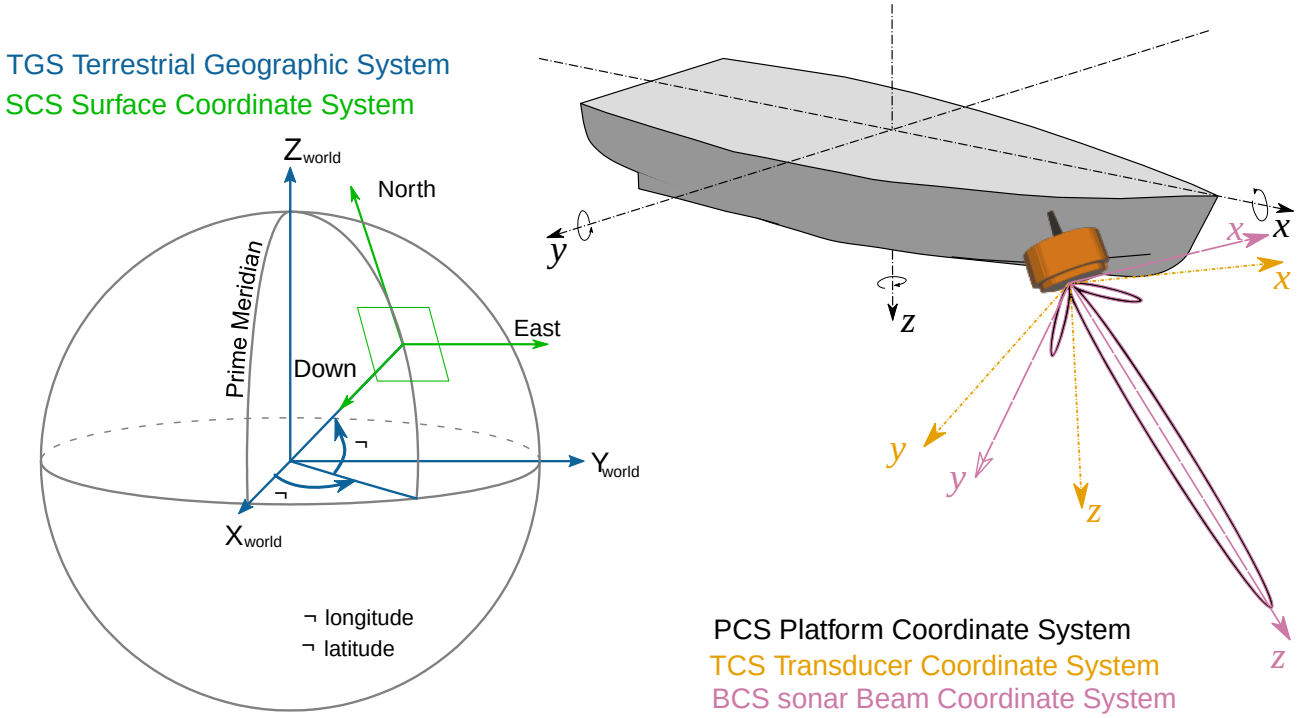

Figure 2. Coordinate systems used to physically position an echo in an absolute geographical system. The arrows on the axes indicate the positive direction and the circular arrows around each platform coordinate system axis indicates positive rotations (Hull drawing based on image from Simrad SN90 manual, redrawn with permission).

5.1. Coordinate System Transforms

Coordinate system transforms $F_{A->B}$ can be expressed as a combination of a Rotation and a Translation of one coordinate system A into another B.

$$F_{A->B} = R_{A->B} + T_{A->B} \qquad (26)$$

Transformation of coordinates from A to B can then be implemented as:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_B = R_{A->B} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix}_A + T_{A->B} \qquad (27)$$

Rotation is specified via three rotation variables and can be implemented via matrix multiplication. As an example, the matrix that will convert a coordinate from coordinate system PCS to coordinate system SCS is defined by:

$$R_{PCS->SCS} = R_z(h) \cdot R_y(p) \cdot R_x(r) \qquad (28)$$

where $R_z(h)$ is the rotation about the z-axis by h degrees, etc. The rotation symbols are chosen to reflect their common nautical names: h for heading, p for pitch, and r for roll and used thus:

[$$\begin{eqnarray}

R_z(h) =
\begin{bmatrix}

\cos(h) & -\sin(h) & 0 \\
\sin(h) & \cos(h) & 0 \\
0 & 0 & 1
\end{bmatrix} \\

R_y(p) =
\begin{bmatrix}
\cos(p) & 0 & \sin(p) \\
0 & 1 & 0 \\
-\sin(p) & 0 & cos(p)
\end{bmatrix} \\

R_x(r) =
\begin{bmatrix}
1 & 0 & 0 \\
0 & \cos(r) & -\sin(r) \\
0 & \sin(r) & \cos(r)
\end{bmatrix}

\end{eqnarray}$$] | *stem-73b500e83e170fbc9388f3c2d26efb77.png*

noting that the order of the matrices is important and should follow this order.

To be clear, the right-hand rule when applied to roll, pitch, and heave means that:

- looking along the positive *x*-axis, a positive rotation (roll) is clockwise (to starboard),
- looking along the positive *y*-axis, a positive rotation (pitch) is clockwise (bow up),
- looking along the positive *z*-axis, a positive rotation (heading) is clockwise (to starboard).

The orientation of the platform is represented using the *z*–*y*'–*x*" Tait-Bryan intrinsic rotation convention (en.wikipedia.org/wiki/Euler_angles), corresponding to heading, pitch, and roll, respectively. Intrinsic means that rotations about the *y*-axis are measured after any rotation about the *z*-axis, and rotations about the *x*-axis are measured after rotations about the *y*-axis (for comparison, extrinsic angles are applied relative to a fixed-platform orientation). This is the most used rotation convention in the maritime field, and the main effect is that roll is measured relative to the plane tilted by the pitch angle.

Translation are linked to a change of system origin, namely from platform origin and the transducer origins. The coordinate system offsets of the transducer are given in the Platform group. These allow for precise specification of the origin of sonar transducer. The offset is a (x,y,z) tuple in the platform coordinate system. Offsets are to be interpreted as a vector that starts at the platform coordinate system origin and ends at the transducer position. For example, an offset of (1, 2, –3) indicates a position that is 1 m toward the bow, 2 m to starboard, and 3 m above the origin of the platform coordinate system.

Some sensors (e.g. the position, attitude sensor) can have their installation offset and angle defined in the Platform group and relative to the platform-coordinate system. Those parameters are usually taken into account by the sensors and the sensor data values are considered to given relatively to

the platform origin (namely roll, pitch should not be compensate for level arms and installation offset except explicitely specified).

5.2. Split-aperture coordinates

Some sonar beams can estimate the arrival angle of echoes (Chapter 6). These angles are given as *minor* and *major* angles in the sonar beam coordinate system (BCS in Figure 2):

- The *z*-axis is always the beam axis with the origin at the receive transducer,
- The *minor* arrival angle (θ) is given by the angle from the *z* axis, measured in the *x*-*z* plane, and
- The *major* arrival angle (ϕ) is given by the angle from the *z* axis measured in the *y*-*z* plane.

Positive minor angles occur in the positive *x* plane and positive major angles occur in the positive *y* plane. Note that this convention does not follow the right-hand-rule for major angles.

For conventional downward-looking echosounders mounted on a ship it is normal that the sonar beam coordinate system is identical to the transducer coordinate system and hence the split-aperture minor angle is oriented alongships with positive angles towards the bow and the split-aperture major angle is oriented athwartships with positive angles to starboard.

5.3. Equations for positioning echoes in a Surface coordinate system

This section provides examples of equations that can be used to position echoes in a geographical coordinate system. These example make some assumptions and approximations :

- Speed of sound is constant and as a consequence sound displacement follows a straight line.
- Sound travel time are supposed to be symetrical, ie the travel time from transmitting position to echo and travel time from echo to receiving position are the same
- Positionning is computed with respect to rx_beam only, meaning we do not make any computation for intersection between tx_beam and rx_beam and taking into account for the ship displacement between emission and reception

Two main categories of system are considered as case studies : system with beam stabilization and pointing angle relative to SCS (typically ME70) and system where beam are not stabilised and beam pointing angles relative to TCS.

More complex equations and algorithm for more precise positioning or for other sensors are out of scope of this section.

Given an echo *E(beam,i)* where *i* is the sample number (from zero to one less than the number of samples) and *beam* the receiving beam considered, several values can be associated with this echo :

- A time stamp given by

$$t_i = ping\_time + \frac{i \cdot sample\_interval + blanking\_interval - sample\_time\_offset}{2} \quad (30)$$

- Its range r_i given by equation \eqref{eq:timeToRange}

5.3.1. Beams are stabilized by the sonar and beam pointing angles are relative to SCS

When beam are stabilized and installation angle offsets are compensated by the sonar their angles are relative to SCS. The sonar will dynamically change beam angles relative to TCS to follow platform movement and maintain a constant beam angle with respect to the SCS. The change from TCS to SCS is composed of a translation from transducer origin to platform origin, taking into account lever arms, and a rotation taking into account platform heading

The position of the echo can be defined as :

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{SCS} = R_{PCS->SCS}(t_i) \cdot T_{TCS->PCS} + R_z(h(t_i)) \cdot R_{BCS->TCS} \cdot \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & cos(\theta) \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ r_i \end{bmatrix} \quad (31)$$

with ϕ and θ the split-aperture coordinates as defined in Section 5.2:

and

$$R_{BCS->TCS} = \begin{bmatrix} \cos(\Psi) & -\sin(\Psi) & 0 \\ \sin(\Psi) & \cos(\Psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos(\Theta) & 0 & \sin(\Theta) \\ 0 & 1 & 0 \\ -\sin(\Theta) & 0 & cos(\Theta) \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\Phi) & -\sin(\Phi) \\ 0 & \sin(\Phi) & \cos(\Phi) \end{bmatrix} \quad (36)$$

where Φ, Θ, Ψ are the beam pointings angles as given in the rx_beam_rotation_phi, rx_beam_rotation_theta, rx_beam_rotation_psi variables of the beam mode subgroups of the sonar group (see Table 11)

and:

$$R_{PCS->SCS}(t_i) = R_z(h(t_i)) \cdot R_y(p(t_i)) \cdot R_x(r(t_i)) \quad (33)$$

where h, p, r are the heading, pitch and roll as given at the time of the ping in the platform_heading, platform_pitch, platform_roll variables of the beam mode subgroups of the sonar group (see Table 11). (*Those variables can also be defined at a higher sample rate in the platform group (see Table 4)*)

and:

$$T_{TCS->PCS} = \begin{bmatrix} transducer\_offset\_x \\ transducer\_offset\_y \\ transducer\_offset\_z \end{bmatrix} \quad (38)$$

where transducer_offset_x, transducer_offset_y and transducer_offset_z are the installation offsets of the transducer associated with the echo, defined in the platform group (see Table 4)

5.3.2. Beams are not stabilized by the sonar and beam pointing angles are relative to TCS

When beam are not stabilized by the sonar, the beam angles are given relative to the TCS and the sonar will not dynamically change the beam angles to follow the platform movement during a

transmit/receive cycle.

The position of the echo can be defined as :

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{SCS} = R_{PCS->SCS}(t_i) \cdot T_{TCS->PCS} + R_{PCS->SCS}(t_i) \cdot R_{TCS->PCS} \cdot R_{BCS->TCS} \cdot \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & cos(\theta) \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ r_i \end{bmatrix} \quad (35)$$

with ϕ and θ the split-aperture coordinates as defined in Section 5.2:

and:

$$R_{BCS->TCS} = \begin{bmatrix} \cos(\Psi) & -\sin(\Psi) & 0 \\ \sin(\Psi) & \cos(\Psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos(\Theta) & 0 & \sin(\Theta) \\ 0 & 1 & 0 \\ -\sin(\Theta) & 0 & cos(\Theta) \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\Phi) & -\sin(\Phi) \\ 0 & \sin(\Phi) & \cos(\Phi) \end{bmatrix} \quad (36)$$

where Φ, Θ, Ψ are the beam pointings angles as given in the rx_beam_rotation_phi, rx_beam_rotation_theta, rx_beam_rotation_psi variables of the beam mode subgroups of the sonar group (see Table 11)

and:

$$R_{TCS->PCS} = R_z(transducer\_rotation\_z) \cdot R_y(transducer\_rotation\_y) \cdot R_x(transducer\_rotation\_x) \quad (37)$$

with transducer_rotation_x, transducer_rotation_y and transducer_rotation_z, the installation angles of the transducer associated with the echo, defined in the platform group (see Table 4) (Note that calibration angles are supposed to be integrated in these variables)

and:

$$T_{TCS->PCS} = \begin{bmatrix} transducer\_offset\_x \\ transducer\_offset\_y \\ transducer\_offset\_z \end{bmatrix} \quad (38)$$

with transducer_offset_x, transducer_offset_y and transducer_offset_z the installation offsets of the transducer associated with the echo, defined in the platform group (see Table 4)

and:

$$R_{PCS->SCS}(t_i) = R_z(h(t_i)) \cdot R_y(p(t_i)) \cdot R_x(r(t_i)) \quad (39)$$

where h, p, r are the heading, pitch and roll as given in the platform_heading, platform_pitch, platform_roll variables of the sonar group (see Table 11) or platform group (see Table 4)

5.4. Equations for positioning echoes in a Geographical Coordinate System

The Surface Coordinate system is a coordinate system oriented with platform heave at the origin of the platform origin. In order to define an absolute position, the coordinate should be converted to a Terrestrial Geographic System coordinate and take into account heave, draft, tide and so on.

The Terrestrial Geographic System coordinate will use latitude and longitude to define its' position

with respect to Earth and elevation will be defined with respect to a absolute surface reference. The absolute surface reference could be the WGS84 ellipsoid, or a surface such as MSL (Mean Sea Level) or LAT (Lowest Astronomical Tide)

Switching from SCS system to TGS is achieved by applying a vertical translation from the platform origin to the surface reference of the TGS The vertical translation at each time is a vector obtained by adding the variables :

- waterline_to_chart_datum matching the distance from the surface reference to the waterline. This variable takes into account the tide and the surface definition for ship or similar
- platform_vertical_offset containing the heave and draft for ship or similar

Each translated coordinate can be then expressed as latitude-longitude coordinates given the platform position retrieved from GPS and the coordinates expressed in SCS system.

Chapter 6. Split-aperture beams

Some sonar beams can estimate the arrival angle of echoes using the split-aperture method and SONAR-netCDF4 supports several ways to store such angles. The type of split-aperture beam data contained in a SONAR-netCDF4 file is given by the *beam_type* variable in each Beam_groupX subgroup. Split-aperture echo arrival angles are always given in the sonar beam coordinate system (see Chapter 5).

When sub-beams from a split-aperture transducer are provided, it is also necessary to combine them to give the power or amplitude of the echoes, as received by the combined beam. All known split-aperture systems achieve this by taking the average of the sub-beam values, noting that the sub-beam data are typically complex power or amplitude values.

To allow for existing conventions on how the split-aperture angles are stored or derived (typically as phase angles between the split-aperture signals), angle sensitivities are given in variables *echoangle_major_sensitivity* and *echoangle_minor_sensitivity*.

6.1. Angles stored in file

The calculation of the echo arrival angle is done by the sonar and the SONAR-netCDF4 file contains either physical arrival angles or phase angles. For the latter, the minor (θ) and major (ϕ) physical arrival angles are obtained by dividing by the minor η_θ and major η_ϕ angle sensitivities respectively. For consistency, when physical arrival angles are stored in the file, η_θ and η_ϕ should still be present and be set to 1.0.

6.2. Angles not stored in file

The sonar does not calculate the echo arrival angle - they are calculated from the split-aperture sub-beams, each of which are stored in the SONAR-netCDF4 files. Several types of sub-beam arrangements are supported (Figure 3) - the angle calculation methods for each type are given in the following sections. The sub-beams are stored as the *subbeam* dimension in the backscatter variables in Table 11, in the same order as labelled in Figure 3.

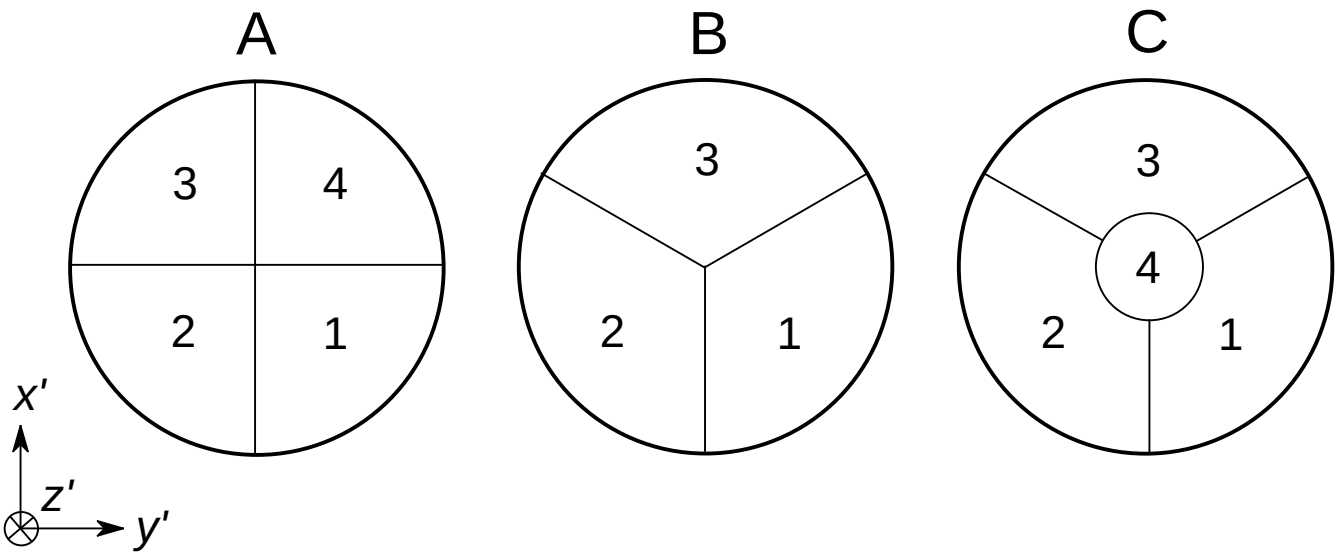

Figure 3. The split-aperture transducer arrangements supported by SONAR-netCDF4 (A: 4 quadrants; B: 3 sub-beams; C; 3+1 sub-beams). The Cartesian coordinates show the orientation of the sonar beam coordinate system relative to the transducer sub-beams (note that the z'-axis points into the page). The numbers indicate the sub-beam labels for each type of transducer.

6.2.1. Four quadrants (type A)

This arrangement has four equal quadrants in the transducer (Figure 3, part A) and the minor (θ) and major (ϕ) angles are calculated thus:

$$y_\theta = (y_3 + y_4) \cdot (y_1 + y_2)^*, \quad (40)$$

$$y_\phi = (y_1 + y_4) \cdot (y_2 + y_3)^*, \quad (41)$$

$$\theta = \frac{\arctan(\Im(y_\theta), \Re(y_\theta))}{\eta_\theta}, \quad (42)$$

$$\phi = \frac{\arctan\left(\Im(y_\phi), \Re(y_\phi)\right)}{\eta_\phi} \quad (43)$$

where $\arctan$ is the four-quadrant inverse tangent, y_x the complex signal from quadrant x, * indicates the complex conjugate, y_θ the phase angle in the minor axis, y_ϕ the phase angle in the major axis, and $\Re$ and $\Im$ the real and imaginary parts of the complex numbered signals. The angle sensitivities are η_ϕ for the major axis and η_θ for the minor axis.

6.2.2. Three or four sub-beams (type B and C)

This arrangement has either three equal sub-beams (Figure 3, part B) or a centre sub-beam and three surrounding sub-beams. (Figure 3, part C). Type C can be treated the same as type B once the centre sub-beam signal is added to each of the outer sub-beam signals. The phase angles between two pairs of the three sub-beams are then given by (Bodholt, 2001; Bjørnø, 2017):

$$\omega_{31} = \arctan\left(\Im(y_3 \cdot y_1^*), \Re(y_3 \cdot y_1^*)\right) \quad (44)$$

$$\omega_{32} = \arctan\left(\Im(y_3 \cdot y_2^*), \Re(y_3 \cdot y_2^*)\right) \quad (45)$$

where $\arctan$ is the four-quadrant inverse tangent, y_x the complex signal from quadrant x, * indicates the complex conjugate and $\Re$ and $\Im$ the real and imaginary parts of the complex numbered signals. The minor (θ) and major (ϕ) angles are then:

$$\theta = \frac{1}{\eta_\theta}(\omega_{32} - \omega_{31}) \qquad (46)$$

$$\phi = \frac{1}{\sqrt{3}\eta_\phi}(\omega_{31} + \omega_{32}) \qquad (47)$$

where η_θ is the angle sensitivity for the minor axis and η_ϕ the angle sensitivity for the major axis.

Chapter 7. Revision history

7.1. Significant changes from version 1 to version 2

Version 2 of this document mades significant changes to the convention presented in version 1. These are summarised here:

decibel units

The reference values for decibel units have been moved from the units attribute to the variable description (typically the `long_name`).

Beam pointing

The method used to specify the beam pointing direction has changed from angles to vectors. A number of coordinate systems and translations and rotations have also been added to enable the flexible and complete definition of beam pointing directions.

Platform attributes

Added variables to the Sonar groups to enable storage of geographic position of the platform interpolated to ping timestamps.

Platform

The platform group can now store data from multiple sensors of the same type.

Platform and Sonar

The transducer description now allows for separate emitting and receiving antennas (as used by some bathymetric sonars). The sonar beam variable names and descriptions have been updated to distinguish between transmit and receive beams.

Equation type

- Type 3 and 4 data have been added in order to handle backscatter data from respectively Simrad EK60 and EK80 echosounders.
- Type 5 data has been added for handling sonars that directly store Sv and/or TS data as defined in MacLennan et al. (2002).
- Type 6 data has been added for handling data recorded by Furuno FCV-38 echosounder.

Beam

- Added variables for the description and storage of data from split-aperture beams via a subbeam dimension added in the Sonar group.
- transmit_frequency_start and transmit_frequency_stop cannot now collapse the beam dimension if all start or stop values are the same (while this increases storage, it will reduce complexity)
- Explicitly allow for type 5 data to have only Sv, only TS, or both in a file.
- added substitute_value_used attribute to some variables to indicate whether the variable has a nominal/default value in it, or an actual measured/calibrated value in it. This is useful for when a value for a mandatory variable is not available.

- add an optional variable to hold the number of samples in each beam, for each ping. This can be used to speed up loading/processing of data files.

Gridded

A new subgroup, Gridded is added to allow for gridded backscatter data (i.e., resampled onto a consistent time/range grid).

ADCP

A new subgroup, ADCP, has been added to store data from acoustic Doppler current profilers.

SingleTarget

A new subgroup, SingleTarget is added to allow for single target detections using backscatter_r and backscatter_i or backscater_r and echoangle_major and echoangle_minor. Calculation of Target Strength (TS) of single targets is given in equation 2,3,4 and 6 for different sonars.

Provenance

Changed the required timestamp format to match that used elsewhere in the convention.

Vendor_specific_SimradWBT

Vendor specific table created with a new subgroup, Vendor_specific_SimradWBT, to provide transducer and transceiver information for processing and reference.

7.2. Changes within a document version

Document version	SONAR-netCDF4 version	Date	Changes
2.0	2.0	April 2022	Significant additions to support additional sonar types.
1.7	1.0	29 May 2018	Modifications due to ICES CRR review and editorial process.
1.6	1.0	7 Feburary 2018	Modifications due to further Topic Group input.
1.5	1.0	20 December 2017	Formatted to ICES CRR style
1.4	1.0	18 September 2017	Added additional CF attributes to time variables. From Topic Group input: incorporation of Type 2 equation and additional variables to support it. Consistency and clarity corrections.
1.3	1.0	21 June 2017	Extensive and significant modifications derived from feedback on v1.2, from creation of test SONAR-HDF4 files, and from implementation of reading in LSSS.
1.2	1.0	2 February 2017	Further modifications after internal review.
1.1	1.0	13 January 2017	Modifications after generation of test datasets.
1.0	1.0	22 December 2016	Draft version for distribution to ICES WGFAST Topic Group on “Defining a data format for omni fisheries sonar”.

Chapter 8. Acknowledgements

We are grateful for the thorough and considered contributions from the Marine Acoustics Society of Japan’s Technical Committee. Advice and experience on the practicalities of reading netCDF files in analysis software was generously provided by Christian Michelsen Research AS, Echoview Software, and Nortek AS. Sindre Vatnehol and Arne Johannes Holmin kindly developed software to generate SONAR-netCDF4 files. We also thank Elodie Fernandez and Jim Biard for their invaluable reviews and suggestions for improving both the convention and this document.

Chapter 9. References

Bjørnø, L. 2017. Applied Underwater Acoustics. Elsevier.

Bodholt, H. 2001. Split-Beam Transducer with 3 Sections. *In* Proceedings of the 24th Scandinavian Symposium on Physical Acoustics, pp. 32–39. Ustaoset, Norway.

Eaton, B., Gregory, J., Drach, B., Taylor, K., Hankin, S., Blower, J., Caron, J., *et al.* 2017. NetCDF Climate and Forecast (CF) Metadata Conventions, Version 1.7.

Furusawa, M., Hamada, M., and Aoyama, C. 1999. Near Range Errors in Sound Scattering Measurements of Fish. Fisheries Science, 65: 109–116.

Gee, L., Doucet, M., Parker, D., Weber, T., and Beaudoin, J. 2012. Is Multibeam Water Column Data Really Worth the Disk Space? *In* Conference Proceedings of Hydro12, pp. 81–86. Hydrographic Society Benelux, Rotterdam, The Netherlands.

ICES. 2016. A Metadata Convention for Processed Acoustic Data from Active Acoustic Systems. SISP 4 TG-AcMeta Version 1.10, ICES WGFAST Topic Group, TG-AcMeta.

ISO. 2014. ISO 19115-1. Geographic Information - Metadata - Part 1:Fundamentals. International Standard.

Jackson, M. A., Groeber, M. A., Uchic, M. D., Rowenhorst, D. J., and Graef, M. D. 2014. H5ebsd: An Archival Data Format for Electron Back-Scatter Diffraction Data Sets. Integrating Materials and Manufacturing Innovation, 3: 1–12.

Macaulay, G. J., Vatnehol, S., Gammelsæter, O. B., Peña, H., and Ona, E. 2016. Practical Calibration of Ship-Mounted Omni-Directional Fisheries Sonars. Methods in Oceanography, 17: 206–220.

MacLennan, D. N. 1986. Time Varied Gain Functions for Pulsed Sonars. Journal of Sound and Vibration, 110: 511–522.

MacLennan, D. N., Fernandes, P., and Dalen, J. 2002. A Consistent Approach to Definitions and Symbols in Fisheries Acoustics. ICES Journal of Marine Science, 59: 365–369.

McQuinn, I. H., Reid, D. G., Berger, L., Diner, N., Heatley, D., Higginbottom, I., Andersen, L. N., *et al.* 2005. Description of the ICES HAC Standard Data Exchange Format, Version 1.60. ICES Cooperative Research Report No. 278.

Nedelec, S., Ainslie, M., Andersson, M., Sei-Him, C., Halvorsen, M., Linné, M., Martin, B., *et al.* 2021. Best Practice Guide for Underwater Particle Motion Measurement for Biological Applications. Technical Report by the University of Exeter for the IOGP Marine Sound and Life Joint Industry Programme. http://dx.doi.org/10.25607/OBP-1726.

Sawada, K., and Furusawa, M. 1993. Precision Calibration of Echo Sounder by Integration of Standard Sphere Echoes. Journal of the Acoustical Society of Japan, 14: 243–249.

Soule, M., Barange, M., Solli, H., and Hampton, I. 1997. Performance of a new phase algorithm for discriminating single and overlapping echoes in a split–beam echosounder. ICES Journal of Marine

Science, 54: 934–938.

The HDF Group. 2017. Hierarchical Data Format, Version 5. http://www.hdfgroup.org/HDF5.

Triton. 2013. Triton Imaging, Inc. eXtended Triton Format (XTF) Revision 35.

UCAR. 2014. UDUNITS. http://www.unidata.ucar.edu/software/udunits.

Unidata. 2017. Network Common Data Form (netCDF) Version 4. UCAR/Unidata.

Annex 1: Working with netCDF4 files

NetCDF4 files are not commonly used for fisheries acoustics data. To facilitate the use of such files, this section provides simple examples of how to access and use SONAR-netCDF4 files in commonly used programming languages, namely Python, R, and Matlab.

Python example

```
# Import packages needed to read and view data from netcdf4
from netCDF4 import Dataset
import numpy as np
import matplotlib.pyplot as plt

# Name of the netcdf file
filename = 'SU90-D20171107-T195023.nc'

# Open the file
dataset = Dataset(filename)

# Open the group where the backscatter data is located
SonarGr = dataset.groups['Sonar'].groups['Beam_group2']

# Get the backscatter data from the 10th ping and and 31st beam
back_r = SonarGr.variables['backscatter_r'][9,30]
back_i = SonarGr.variables['backscatter_i'][9,30]

# Close dataset
dataset.close()

# Compute the power
power = abs(np.vectorize(complex)(back_r,back_i))**2

# Plot the power of beam
plt.figure(1)
plt.clf()
plt.plot(10*np.log10(power))
plt.xlabel('Sample number')
plt.ylabel('Amplitude [dB]')
```

R example

```
library(h5)
SU90_nc <- PATH_TO_SONAR_NETCDF4_FILE

# The h5 package is able to read the example file:
data <- h5file(SU90_nc)

# Show the contents:
data
```

```
# List the dimensions of the data sets:
ds <- list.datasets(data)
dims <- lapply(ds, function(x) openDataSet(data, x)@dim)
names(dims) <- ds
dims

# Note that the backscatter data are stored as variable
# length datatype, so the dimensions in 'dims' does not
# reflect to the full size of the backscatter data.
# In this particular file the length of all beams and all
# pings of both Beam_group1 and Beam_group2 are identical:

bs_r1_name <- "/Sonar/Beam_group2/backscatter_r"
bs_i1_name <- "/Sonar/Beam_group2/backscatter_i"

# Save the real part and set dimension to
# [length of beams, number of beams, number of pings]:
backscatter1 <- complex(
real=unlist(data[bs_r1_name][10,31]),
imaginary=unlist(data[bs_i1_name][10,31]))
power <- abs(backscatter1)^2

# Plot the first and second beam of the third ping:
plot(10*log10(power), type="l",
ylab="Power", xlab="Sample number")

# Close the file:
h5close(data)
```

Matlab example

```
% Example Matlab script to load a SONAR-netCDF4 file using the
% high-level HDF5 functions.

% File to read
file = 'SU90-D20171107-T195023.nc';

% Ping and beam to read
pingNo = 10;
beamNo = 31;

% Read the selected ping/beam from the file
amp_r = h5read(file, '/Sonar/Beam_group2/backscatter_r', [beamNo pingNo], [1 1]);
amp_i = h5read(file, '/Sonar/Beam_group2/backscatter_i', [beamNo pingNo], [1 1]);

% Calculate the power values for the ping.
power = abs(complex(cell2mat(amp_r), cell2mat(amp_i))).^2;

% Plot the power
plot(10*log10(power))
```

```
xlabel('Sample number')
ylabel('Power (dB)')
```