HUDDL for description and archive of hydrographic binary data

Giuseppe Masetti and Brian Calder

Center for Coastal and Ocean Mapping & Joint Hydrographic Center – University of New Hampshire (USA)

Abstract

Many of the attempts to introduce a universal hydrographic binary data format have failed or have been only partially successful. In essence, this is because such formats either have to simplify the data to such an extent that they only support the lowest common subset of all the formats covered, or they attempt to be a superset of all formats and quickly become cumbersome. Neither choice works well in practice. This paper presents a different approach: a standardized description of (past, present, and future) data formats using the Hydrographic Universal Data Description Language (HUDDL), a descriptive language implemented using the Extensible Markup Language (XML). That is, XML is used to provide a structural and physical description of a data format, rather than the content of a particular file. Done correctly, this opens the possibility of automatically generating both multi-language data parsers and documentation for format specification based on their HUDDL descriptions, as well as providing easy version control of them. This solution also provides a powerful approach for archiving a structural description of data along with the data, so that binary data will be easy to access in the future. Intending to provide a relatively low-effort solution to index the wide range of existing formats, we suggest the creation of a catalogue of format descriptions, each of them capturing the logical and physical specifications for a given data format (with its subsequent upgrades). A C/C++ parser code generator is used as an example prototype of one of the possible advantages of the adoption of such a hydrographic data format catalogue.

Introduction

The hydrographic field is characterized by the presence of a large number of binary data formats using to acquire acoustic data. In fact, almost every manufacturer has its own formats (e.g., Kongsberg Simrad, Ixsea, L3 Klein, SAIC, Triton Imaging). Each format in use usually has a long history with many different releases; and, in the future, there is no reason to think that there will not be more novel data formats or updated releases of the existing formats to manage additional features or innovative systems. Although some formats are more commonly used as exchange formats than others as, e.g., the Generic Sensor Format (GSF) or the Extended Triton Format (XTF), there is not a *de facto* standard format.

Data parsing, a preliminary step for all applications, represents a common speed bump both for researchers and students involved in such fields as marine geology, habitat mapping, ocean engineering, etc. That is, having to deal with a new data format involves all of the effort required

to build a reader and/or writer for the new data, even if that data is only used once. This acts as a disincentive to use data, and is also an unwarranted distraction for researchers from their primary goal of understanding the data itself.

At the same time, there is a lack of a common repository for all the format specifications. It is therefore often required to dig into different manufacturer websites in order to obtain the last release of a given data format, and this is often driven by issues experienced after having attempted to read new survey lines. Furthermore, data parsers or tools to convert among different data formats are only available (some for free, the largest part with a cost) for a limited number of them. One of the most relevant (and negative) consequences of the current situation is that everybody has to write their own parsers (mostly from scratch) for every data format in which they are interested, and then keep them up to date: that is a lot of boring and error prone work (that is not strictly necessary), which can lead to variant interpretations of the same data in different software packages.

A concomitant assessment required by this topic comes from the need to access legacy data formats (e.g., for historic trend evaluation). The optimal solution to the overall data formats issue should also be able to offer access to that type of data, providing a mechanism to describe data collected and archived in sometimes 'exotic' data formats (e.g., developed by now defunct manufacturers). Solving this issue implies the definition of a reliable way to access the data collected today by our descendants, with evident advantages in the adoption of these methods by hydrographic data archiving centers.

Many of the past attempts to introduce a universal hydrographic binary data format have failed or have been only partially successful. In essence, this is because such formats either have to simplify the data to such an extent that they only support the lowest common subset of all the formats covered, or they attempt to be a superset of all formats and quickly become cumbersome. Neither choice works well in practice: the first approach tends to simplify too much the semantic of data collected (leaving out important manufacturer specific information), the second becomes more and more unwieldy and cumbersome as the number of supported formats increase.

In our opinion, a better long-term solution to the data format issue is represented not by a new attempt to create a chimeric new data format, but by the creation of a descriptive language able to describe all the past, the existing, and the future hydrographic data formats. We call this idea the Hydrographic Universal Data Description Language (HUDDL). A simple and low-cost metadata link to an online HUDDL repository of data format descriptions could provide a robust way to describe how the data are organized within the binary data. At the same time, HUDDL could drastically reduce the time required in accessing hydrographic binary data, so that researchers and software developers can focus more on development of new algorithms and less in the boring, but mandatory, data parsing step.

The main difference of such an approach with respect previously proposed solutions based on a chimeric data format is in the idea to have a system that adapts code to the data to be accessed rather than trying to adapt the data to the code. If we are able to describe in a machine-readable way the content of a data format, this description can then be used in multiple ways. For example, such a description could be used for:

- Automatic generation of data readers and writers.
- Validation of the content of survey lines claiming to be consistent with a particular format release.
- Easy recovery of partial information from corrupted data.
- Storage and reference of the description of how data are organized in a given data format.
- Incremental update of data format specifications.
- A mechanism to uniformly and consistently produce documentation for different formats.

In addition, the same description mechanism represents a tool for definition and testing of new data formats or updates before being released to the hydrographic community.

After a brief discussion on the rationale of HUDDL and related existing works, the present paper provides some details for a conceptual model and for a physical implementation, and it concludes with potential implications of wide adoption of the HUDDL framework.

Rationale

Data format conversion is the soft underbelly of processing scientific data, consuming immense amounts of time for those who work in a heterogeneous software environment (Georgieva et al., 2009). From this point of view, this project represents a way to reduce, in a relatively short time, existing problems related to data interoperability and data access for marine scientists, issues that too often divert attention from algorithm development and testing.

As a primary contribution to the solution of the above problem, HUDDL provides a technique and a prototype implementation to allow the description by schemas of the physical representation and the overall structure of various existing binary files used in the hydrographic field. Such a system can:

- Provide a common set of many basic validation and computation functionalities.
- Explain the structure of binary files to users (readability).
- Automatically generate a parser directly from a schema.
- Provide a convenient basis for building arbitrary transformations between binary data formats (data conversion/transformation) and data file indexing.
- Extend applications with content-aware functionalities (e.g., tools that can inspect any binary file given a schema, file comparison, etc.).

All of the above features motivate our interest in such a solution.

Although the approach is theoretically applicable to any generic and arbitrary binary format, this project is focused on describing most types of hydrographic binary data formats in a simple syntax. The main reason for this is the desire to maintain implementation libraries that are lightweight and as simple as possible, providing order rather than adding complexity in the already extensive sea of data formats.

At the same time, this solution provides an inexpensive but robust way to deal with legacy data formats. When required to access old datasets in an arbitrary binary format, an *ad hoc* XML schema may be created that can deal with the specific vagueness of some legacy formats or some rare variant implementations.

Another peculiarity of HUDDL is that it provides a powerful tool for data archiving centers that store data in their native formats (rather than converting them to a – rarely existing – standard format). Since the creation of a HUDDL description schema describing a data format requires minimal effort in terms of time and computer science knowledge, legacy datasets may acquire a renewed accessibility, and scientists will in the future be able to efficiently and easily retrieve data (e.g., for studies on historic trends of geologic features).

Existing related works

The problem of describing binary data files, and translating them, is relatively common, and there have been a number of different approaches to it over the last 30-40 years. For example:

- Data Extraction, Processing and REStructuring System (EXPRESS), developed by IBM at the end of 1970's, supports access to a wide variety of data and restructuring of it for new uses. The system was driven by two very high level nonprocedural languages: DEFINE for data description and CONVERT for data restructuring (Lum et al., 1976; Shu et al., 1977).
- STandard for the Exchange of Product model data (STEP), the unofficial name for the evolving ISO standard 10303 - Product Data Representation and Exchange. The standardization began in 1984, and the first set of International Standard documents was approved in 1994. Although the standardization effort was much wider, this standard is mainly used by industry to facilitate data/information exchange between different computer aided design (CAD) systems and downstream application systems. ISO 10303 exchanges usually employ the neutral file approach, in which transfer between two systems is a two-stage process (Pratt, 2001). Firstly data are translated from the native data format into the neutral ISO 10303 format (the exchange medium may be an ASCII file). Then, a translation from the neutral format into the native format of the receiving system is applied. The entities to be captured and exchanged using STEP, and their relationships, are defined in schemas written in EXPRESS (Pratt, 2001; Schenck and Wilson, 1994).

- External Data Representation (XDR), an IETF standard (RFC4506) since 1985 that uses a base unit of 4 bytes, serialized in big-endian order, so that smaller data types still occupy four bytes each after encoding. Its main use is for the description and encoding of data in binary files to be used by a computer network protocol (Eisler, 2006). This, as with the previous projects, is not XML-aware.
- The Hierarchical Data Format (HDF) project, run by the National Center for Supercomputing Applications (NCSA). It involves the development and support of software and file formats for scientific data management. The HDF software includes I/O libraries and tools for analyzing, visualizing, and converting scientific data. HDF also, however, defines a binary data format in which the data is represented (Folk et al., 2011), so it cannot be used for general binary data formats. HDF also provides software that allows the conversion of (most) HDF files to a standard XML representation, and HDF5 adopts a validity check mechanism based on XML Schema and XQuery for evaluating constraints satisfaction (Folk et al., 2011).
- Data Format Description Language (DFDL) and BinX. DFDL, a product of the Global Grid Forum, is a language for describing text and binary data formats. DFDL describes all aspects of a data format, from byte-order to structure. DFDL is derived from BinX, which concentrates on binary data formats (Powell et al., 2011; Westhead and Bull, 2003). By mapping binary formats to XML Schema, DFDL can be used to parse binary data into an XML-based data model (Futrelle and McGrath, 2011).
- ESML, a language for describing binary data used in the earth sciences. This allows applications to access these formats using a common interface, thereby simplifying and speeding application development (Ramachandran et al., 2004).
- Protocol Buffers, a technique developed by Google for representing data in their Intranet infrastructure and open-sourced in 2008. It allows the definition of simple data structures in a special (not standardized) definition language (Varda, 2008). The claims made by Google regarding the efficiency and effectiveness have only be proved to some extent (Kaur and Fuad, 2010), and right now it does not seems to being adopted as a *de facto* industry standard.
- Java-Script Object Notation (JSON) is an open standard format that can be readily mapped to object-oriented systems and which is much simpler than XML. The future JSON Schema will add the possibility to validate data, but it is still an internet draft. JSON is not a fully mature technology and is just becoming more widely known.

The last two works mostly network-oriented Interface Definition Languages used to transmit data between a server and web applications (and in this field are competitors of XML). They have very limited or missing display capabilities because they have not been developed as document exchange formats. For instance, Protocol Buffers has the limitation that there should not be more than one element with the same name to satisfy its specification on naming message fields (Kaur and Fuad, 2010). However, since HUDDL description schemas are well-formed and valid XML files, existing algorithms can be used to map HUDDL schemas into, for instance,

JSON format or Protocol Buffers interface files once (and if) these technologies become mature and safer in the future.

None of the above alternatives provides the full set of requirements for an XML-based catalogue of hydrographic data formats. In essence, the system must be:

- Easily adopted (for instance, since the adoption of an XML-based syntax presents all the previously discussed advantages, this requirement is missed by the first solutions).
- Well-maintained (some techniques, e.g., ESML, do not have any time schedule for standard development).
- Widely accepted (there is a common lack of this requirement in any of the existing solutions, which may represent a hint of weakness in the adopted implementation).
- Flexible and with a low-cost implementation (HDF requires data conversion in another binary format, while manufacturers likely want to maintain their own data formats).
- Based on a sufficiently simple syntax (the intent of DFDL to be universal increases the overall complexity), while still retaining enough expressivity to describe hydrographic binary data formats.
- Available with an open-source and open-community implementation (the hydrographic community is narrower than the communities targeted by each existing solution, which may speed up the adoption and the contribution to develop a working approach). For instance, Protocol Buffers, while open source, is not open in the development process. Although everybody is free to file issues, development is largely controlled and guided by Google.

Furthermore, none of the proposed methods explicitly focus on hydrographic use cases, which are dominated by data streams of sensor data, and arrays and lists of floating point numbers.

Although none of the proposed solutions are completely adequate, features from each of the existing solutions do have merit and have been adopted into HUDDL.

Conceptual model

A community-specific descriptive language such as HUDDL is simplified at a certain level by the fact that all the existing data formats represent different answers to the same problem: fast storage of data acquired in real-time.

All the evaluated data formats have three components (Figure 1):

- The semantic; that is: what a given value collected in the data format actually means (e.g., the unit of measure).
- The physical description; how the bits and the bytes are stored on disk (e.g., endianess, memory alignment).

- The logical structure; that is, what data structures are used to organize the data (e.g., array).

These components represent three pieces of the puzzle required to describe a (hydrographic) data format. We will first describe each of these pieces at a conceptual level, then a possible physical implementation is provided in the next section.

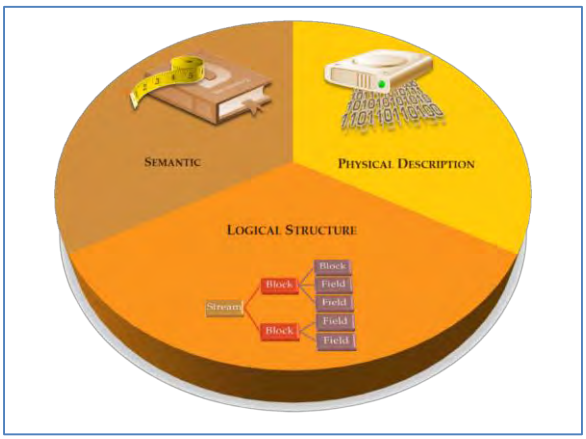

Figure 1 - Threefold nature of a hydrographic data acquisition format.

The study of the threefold nature of any data format directly drives toward quite a natively tree-structured conceptual model: a top-level container, called a 'Schema' that may hold several different descriptions of data formats, each of which has both a 'Prolog' and a 'Content' element (Figure 2).

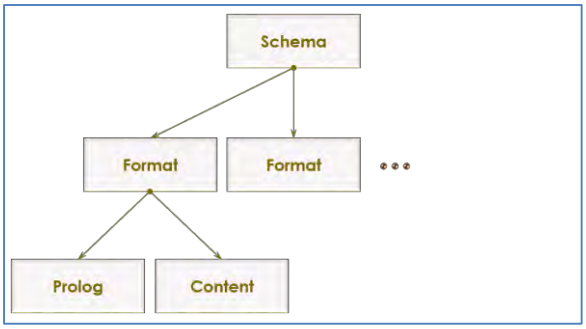

Figure 2 - Top-level elements of the HUDDL conceptual model.

The 'Prolog' represents a collection of metadata related to the described data format, such as the organization that created it, the personnel responsible for its maintenance, or the history of releases (Figure 3). As described later in the paper, this information will be used in the creation of homogeneous and consistent documentation for different data formats.

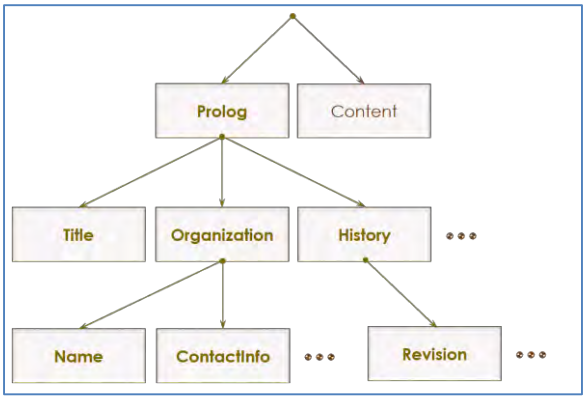

Figure 3 - Example of elements present in the 'Prolog' branch of the HUDDL conceptual model.

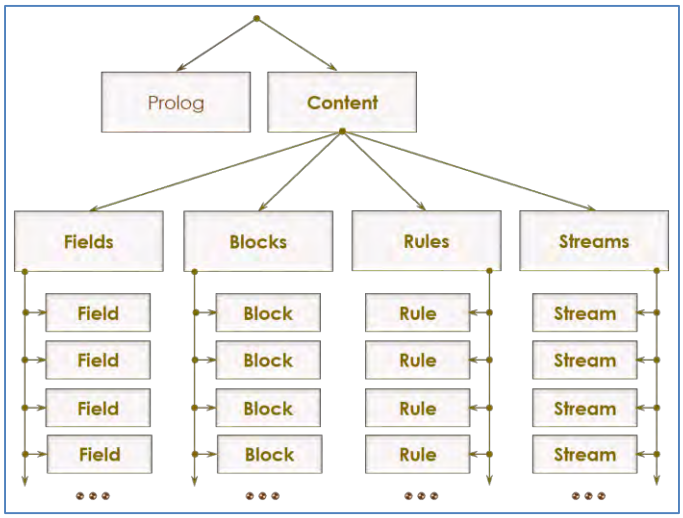

Figure 4 - Four sections of the 'Content' branch of the HUDDL conceptual model.

The HUDDL 'Content' branch is used to describe both the structure and the format of a binary data file in a platform-independent way (Figure 4). This branch has four main types of containers:

- Fields, used as basic value containers (e.g., bytes, two- or four-byte integers, floating point numbers, etc.).
- Blocks, which may contain any number of fields, other (nested) blocks, and available data structures (e.g., 2D array). A block represents logically related group of information elements committed to file at the same time instant.
- Rules, for explicitly describing relations among fields and other available data structures.
- Streams, each of which represents the top-level main data structure for a given release of a data format. Each stream contains the description of the overall composition of a data file. Thus, it is possible to have in the same document multiple streams that reflect different versions of the same data format. Hence updates are only required to be incremental (this

characteristic makes them smaller, simpler, and faster than would be required for a single-release format description approach).

There is a wide range of possible selection for the physical implementation of the conceptual model just outlined. Among them, an Extensible Markup Language (XML) solution has been selected (at the present time) as discussed in the next section.

Physical implementation

The XML solution

XML, with the support of strictly linked XML Schemas, provides a representation standard that is convenient for many reasons:

- It is both human- and machine-readable, easily and quickly extensible.
- It has wide adoption.
- It is a mature technology.

Though not strictly required by web services standards, XML is also the *de facto* standard wire format for web services (Chiu et al., 2005), and therefore holds a unique, pervasive role in that area. Although it could be tempting to think about a future where all data are exchanged in some forms of XML-like format, use of XML for data has many disadvantages and offers very few opportunities for many fields of science and engineering (Westhead and Bull, 2003).

The same disadvantages are valid for hydrographic data sets, which are usually based on large (often hundreds of GBs) and regularly structured binary files, with existing tools for data reading and manipulation often written in languages (e.g., Fortran, C) with primitive file handling capabilities. Rewriting these tools to be XML-aware as suggested by some researchers (Chiu et al., 2005) would require significant efforts and have a number of drawbacks. For instance, read/write operations and data transfer (since the XML representation is larger than the binary) would take significantly longer, XML is very verbose, the computational costs of converting floating-point numbers from ASCII to a native machine representation (such as IEEE 754 (2008)) can be significant, and some inappropriate/inefficient data structure representations can lead to delays (e.g., the proposed standard XML representation for a multidimensional array is a tree of lists, while the popular Simple Object Access Protocol (SOAP) data model defines arrays to be a parent elements containing a sequence of child elements). It is thus unlikely that, at least in the medium term, sonar manufacturers will follow this path, nor will hydrographic data collectors push in this direction.

XML does (and will have) a key role in the representation of metadata associated with an hydrographic dataset, however, since it represents the canonical means to describe information

related to the data collector, acquisition parameters, meteorological conditions, etc. At the same time, hydrographic applications that once were tightly-coupled and monolithic are now becoming more modular, with collaborating components spread across diverse computational elements (Calder, 2013). In such a distributed environment, open metadata systems are increasingly important and useful to communicate through the exchange of substantial amounts of structured data (Widener et al., 2001). The increasing popularity of XML in the field of marine science and engineering is also driven by its relevant role in the ISO 19000 series metadata standard (Georgieva et al., 2009; Hua and Weiss, 2011; ISO, 2008; Yongguo et al., 2009).

This paper explores the application of XML technology for a standardized description of (past, present and future) binary acquisition data formats in use in the hydrographic field. That is, the XML data is used to give a structural description of the contents of a file format, rather than the content of a particular file. Done correctly, this opens the possibility of automatically generating the code to read/write a file format from the XML description (in any of a number of different languages), as well as automatically generating documentation on the content of the file format, and providing easy version control of file formats. At the same time, coupling one or more of the proposed descriptive XML schemas with a given hydrographic dataset, as metadata, represents a detailed definition on how data have been actually stored.

HUDDL has the simple aim of proposing a relatively low-effort solution that may harmonize and catalogue the wide (and sometimes wild) range of hydrographic data formats, with their multiple revisions and releases. In a nutshell, the proposed solution is to provide a catalogue of HUDDL format descriptions written in XML, each of them containing description for a given data format (with its subsequent upgrades) that can be used as a set of instructions for an application on how to manipulate a data file in a specific format/version.

HUDDL is an XML-based language that is used to describe intricate 'existing' hydrographic data format specifications in a relatively concise manner. It is not a programming language, nor a protocol (hence it does not inherently give rise to any particular security considerations). XML was selected, rather than create a language from scratch (Georgieva et al., 2009), because of the wide variety of tools that are available. The HUDDL format descriptions are based on the HUDDL core schemas (which are also XML schemas) so that they can be created – using any of the available XML off-the-shelf tools – to both describe the physical and logical implementation of a data format, as well as to create documentation to explain the semantic meaning to a human. These tools are intelligent, not allowing invalid data entry, and suggesting that which is valid.

Programs can read HUDDL descriptive schemas through any of a variety of parsers. Some XML parsers are already built into programming languages (e.g., Java, Python), and there are a variety of external parsers (e.g., Xerces, libxml). Since they are XML documents, HUDDL schemas can be easily translated to other formats using XSLT (Extensible Stylesheet Language Transformations) translators, which are widely available. The latest generation of web browsers

is able to use XSLT stylesheets directly, so that XML documents can be viewed easily by a human (e.g., HTML pages, PDF files), as well as being understandable by machines.

Since HUDDL format descriptions are external to the data files, they can be easily created, modified and viewed in any text editor. Similarly, a new module will be easily added to the input/output (IO) library in case of new data formats.

From the point of view of software manufacturers, developers will have a new tool to be able to build applications that are more data format independent. Ideally, a single reader component could be developed in isolation and then these modules combined for all the various data formats. This solution will reduce, at least in part, the efforts required to maintain a set of data readers, usually one for each different format, during subsequent updates to the format.

In addition, the creation of a unique Web repository for HUDDL format descriptions could provide a safe and easy-to-check common point for data format specification, and widely used systems (e.g., RSS, or an open-subscription mailing list) could assist in staying current with the last release of data formats for all of the interested players.

An example of a HUDDL format schema

As a proof of concept, this sub-section presents a stub of a HUDDL format schema based on an existing data format (in this case, for a Kongsberg Simrad EM Series data file).

The HUDDL format description has a 'schema' as root element (Figure 5). This top-level element may contains an infinite number of 'format' elements, each of them used to provide the description of a hydrographic binary data format. The description specific to a given data format is provided using two main containers: the 'prolog' for generic information (metadata) about the format (Figure 6), and the 'content' with the actual format specifications.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
<huddl:schema xmlns:huddl="http://huddl.ccom.unh.edu" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    huddl:version="1.0" xml:lang="EN" xsi:schemaLocation="http://huddl.ccom.unh.edu huddl.xsd">
    <huddl:documentation>
        <huddl:note>Testing file for HUDDL 1.0 containing a format description for Kongsberg EM Seriest.</huddl:note>
    </huddl:documentation>

    <!-- example of a format specification -->
    <huddl:format huddl:name="Kongsberg EM Series">

        <!-- example of a format prolog, with generic human-readable infomation -->
        <huddl:prolog>

        <!-- example of a format content description -->
        <huddl:content>

    </huddl:format>

</huddl:schema>
```

Figure 5 - Top-level content of a HUDDL format schema: the schema element may contains one or more format elements, and each of them has a prolog (for metadata information) and a content (for data description).

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
<huddl:schema xmlns:huddl="http://huddl.ccom.unh.edu" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  huddl:version="1.0" xml:lang="EN" xsi:schemaLocation="http://huddl.ccom.unh.edu huddl.xsd">
  <huddl:documentation>
    <huddl:note>Testing file for HUDDL 1.0 containing a format description for Kongsberg EM Seriest.</huddl:note>
  </huddl:documentation>

  <!-- example of a format specification -->
  <huddl:format huddl:name="Kongsberg EM Series">

    <!-- example of a format prolog, with generic human-readable infomation -->
    <huddl:prolog>
      <huddl:title>Instruction Manual - EM Series - Multibeam echo sounders - Datagram formats</huddl:title>
      <huddl:alternateTitle>Kongsberg EM Series formats specification</huddl:alternateTitle>
      <huddl:documentNumber>850-160692/Q</huddl:documentNumber>
      <huddl:date>2013-02-01</huddl:date>
      <huddl:organization>
      <huddl:history>
      <huddl:copyright>
      <huddl:disclaimer>
      <huddl:warning>
      <huddl:supportInfo>
      <huddl:generalNotes>
      <huddl:verboseDescription>
    </huddl:prolog>

    <!-- example of a format content description -->
    <huddl:content>

  </huddl:format>

</huddl:schema>
```

Figure 6 – Example of a prolog for a HUDDL format schema with some or the possible XML metadata tags.

In the 'content' element, we find both the 'blocks' (Figure 7) that are the data containers usually used to carry specific types of information (e.g., navigation, attitude, imagery), and the 'streams' (Figure 8) that are the top-level element of a data format listing all the available/possible 'blocks' for a given release of the described data format.

```
<!-- example of a format content description -->
<huddl:content>
  <huddl:fields>
  <huddl:blocks>

    <huddl:block huddl:name="emhdr">
      <huddl:blockContent>
        <huddl:field huddl:name="datasize" huddl:type="huddl:u32">
          <huddl:intRange>
        </huddl:field>
        <huddl:field huddl:name="stx" huddl:type="huddl:u8">
        <huddl:field huddl:name="type" huddl:type="huddl:u8"/>
        <huddl:field huddl:name="modelnum" huddl:type="huddl:u16"/>
        <huddl:field huddl:name="date" huddl:type="huddl:u32"/>
        <huddl:field huddl:name="time" huddl:type="huddl:u32">
      </huddl:blockContent>
    </huddl:block>

    <huddl:block huddl:name="em96db_revPreA" huddl:padding="notUsed">
    <huddl:block huddl:name="em96depth_revPreA">
      ... more block definitions here ...

  </huddl:blocks>
  <huddl:rules>
  <huddl:streams>
</huddl:content>
```

Figure 7 - An instance of HUDDL 'block' containing the description of a Kongsberg Simrad EM Series datagram.

The data content description that is specific for a given format release/revision is encapsulated within a single ‘stream’. The same ‘block’ can be referenced from multiple releases (streams) so that the description verbosity is reduced and incremental updates are supported.

```
<!-- example of a format content description -->
<huddl:content>
  <huddl:fields>
  <huddl:blocks>
  <huddl:rules>
  <huddl:streams>
    <huddl:stream huddl:revID="preA">
      <huddl:header>
      <huddl:topBlocks>
        <huddl:refBlock>em96depth_revPreA</huddl:refBlock>
        <huddl:refBlock>em96clock</huddl:refBlock>
                ... more blocks here ...
      </huddl:topBlocks>
      <huddl:tail>
    </huddl:stream>
    <huddl:stream huddl:revID="revA">
      ... more revision streams here ...
  </huddl:streams>
</huddl:content>
```

Figure 8 - A 'stream' stub that refers to previously defined 'blocks'.

While simple, this description format is expressive. For example, since blocks can contain other blocks and arrays of blocks, a data unit which contains a header segment (e.g., the parameters for a given ping’s depth detections) along with a record of the depths detected per beam, can be easily represented by block for each detection, and a block that contains the header information as elemental fields, and a 1D array of the detection blocks. The schemas also allow for variable length arrays (e.g., if the number of beams reported is variable per ping), for two dimensional arrays of fields or blocks, and other common features of typical hydrographic data formats. It is therefore typically a fairly simple matter to translate a given data format into a HUDDL description given the appropriate documentation.

HUDDLer: a multi-language generator of hydrographic data readers

Of the many applications of a HUDDL schema description for a data format, one of the most useful is the automatic generation of a library to read and write data in the described format. HUDDLer is a working prototype of a multi-language code generator based on HUDDL schemas, written in C/C++, and providing an IO library.

HUDDLer implements the HUDDL-philosophy of constraining the description of the data format to the schema, so that the user has to touch the minimal amount of code to reflect any change in the data format specification. That is, instead of having to change the user’s application code directly to reflect the format changes, changes to the schema will be translated automatically by HUDDLer into the IO library that represents that data format, and this can be readily automated in most software build systems. In practice, changing to a new data format should be as simple as changing the schema and then recompiling the library or application, as

appropriate, leaving the programmer to work on the application logic to use the new facilities added by the new version of the format.

HUDDLer's workflow is based on a chain of three agents: the HUDDL schema parser, the streaming serializer, and the back-end language-specific generators.

HUDDL schema parser

XML is specified as a textual format for structured data. Beyond its syntax, however, the specification also implicitly defines an abstract data model for tree structured data consisting of parent nodes and child nodes, with their associated attribute, content, and other information. XML Schema is a powerful type language that defines type systems for XML by enabling the precise specification of sets of XML documents, thus facilitating interoperability between producers and consumers of XML documents (Chiu et al., 2005).

The HUDDL Core Schemas added some atomic, typed values to the current XML infoset. This allows the HUDDL Format Schemas to easily describe numbers in their native, machine form, rather than using a more verbose structure.

Streaming serializer

ReadData is a streaming serializer that is suitable for situations where the data is composed of large, complex, and dynamic data structures. It does not impose any high-level structure. Any such structure is implemented using a HUDDL Format Schema as a layer operating on top of ReadData.

The ReadData format is minimalistic, and it supports 1-, 2-, 4-, and 8-byte integers; 4- and 8-byte floating-point numbers, and 1- and 2-dimensional arrays. Both little-endian and big-endian formats are supported. Additional endianesses (although expected to be rare, e.g., bitwise endian formats) will be added as needed.

Back-end language-specific generators

With the adoption of XML schemas to describe the required object-oriented (O-O) output (class templating), the porting of the generated parser code to other popular programming languages (e.g., Java, Python, Octave/Matlab) is straightforward.

Generating code directly in the target language allows the code generator to take advantage of particular language features that would simplify the generated code, or better express the idiomatic nature of the target language usage.

At the same time, the output code from these generators attempts to provide data types that are as transparent as possible in order to reduce the complexity of manipulating routines in the master application.

Framework overview

Due to the extensive implications of the adoption of the HUDDL approach, this section attempts to summarize the main expected benefits.

HUDDL is based on a set of XML Schemas that represents the building blocks that are used to describe the hydrographic data formats:

- Core schema for the machine representation (e.g., endianess, bit-ordering).
- Core schema that is a dictionary of primitive types (e.g., IEEE floats, integer, strings).
- Core schema with data structures (e.g., enumerations, list of fields, array, offset structures, bitfields, bitmasks, etc.).

We propose the creation of an online repository that will contain all of the format schemas based on these HUDDL core schemas (Figure 9). As an alternative, this repository could also be distributed, so that each manufacturer has its own repository, so long as the format schemas are based on the common set of core schemas.

These publicly available HUDDL format schemas may be used as ‘trusted’ references to archived data. As long as a binary data file is paired with a HUDDL schema, the data content is described and the information can be recovered. The main benefit of this is that this makes it much more likely that users will be able to read the data in the future, and have adequate documentation. And all for the price of a metadata link.

A HUDDL format schema (.hdl) can optionally contain a URL which points to files in a folder structure that follow the described encoding. This solution is totally non-invasive for existing data formats. However, in a future, data formats could include a field with a data-format unique identifier released by the Web repository.

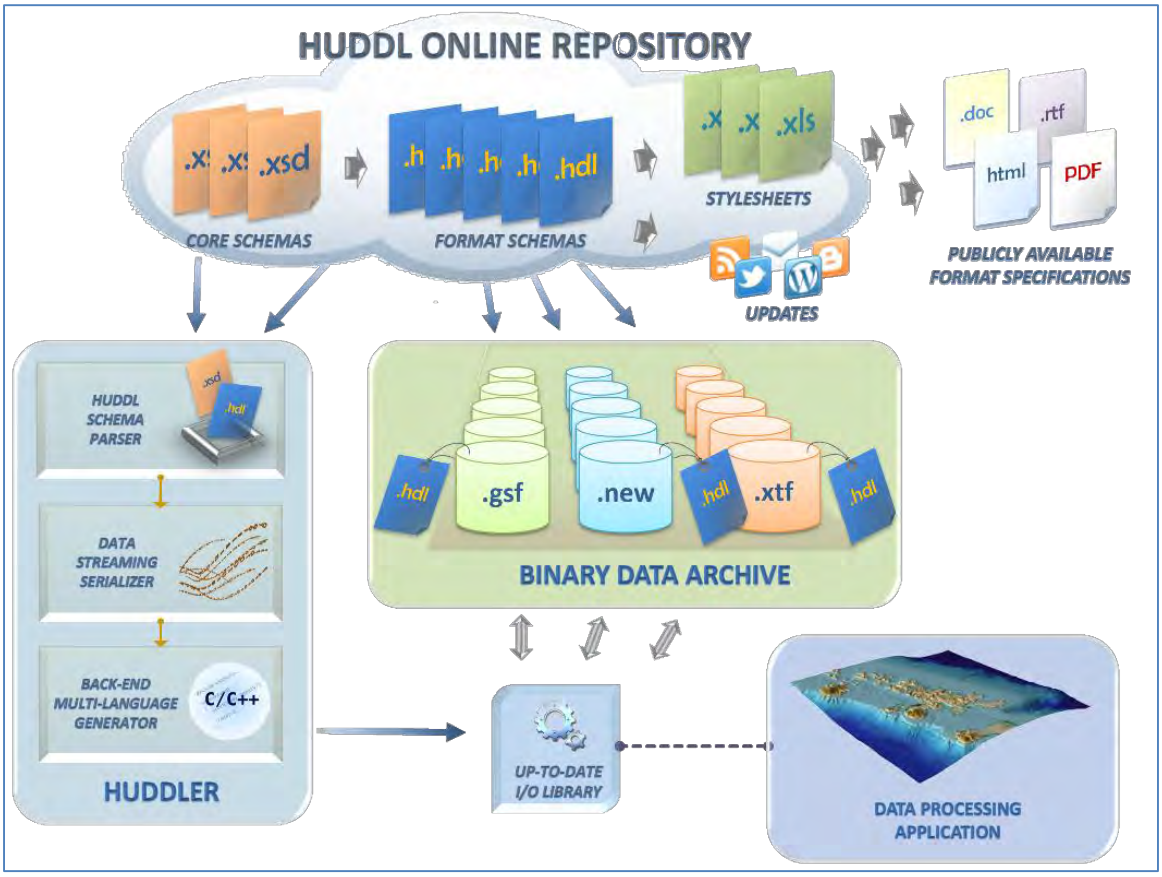

Figure 9 - HUDDL framework: the online repository is used both for publicly providing format specifications (in different formats) and as source for code generators (in figure, the prototype HUDDLer) that parse the descriptive schemas, serialize the information and create an I/O library. Data processing applications can thus relay on this library for access the binary data.

Machine data representations are becoming much more standard than previously. In particular, the prevalence of the IEEE floating point standard has simplified a number of the issues (IEEE, 2008). For the physical representation it is still necessary to specify:

- The byte ordering: big-endian or little-endian (with the possibility to extend to other less common, or historically used, endianess, e.g., middle-endian of PDP-11 series).
- The bit ordering: big-endian or little-endian (almost always big-endian).
- Block size: an approach common for many binary formats is padding out data fields so that they are always a multiple of a given block size.

HUDDL follows the same assumption present in the XDR standard: bytes (or octets) are portable, where a byte is defined as eight bits of data (Eisler, 2006). The outcome of this is that a given hardware device encodes the bytes onto the various media in such a way that other hardware devices may decode the bytes without loss of meaning.

As primitive types, the project provides a broad range of options, including all the types that are already represented in XML Schemas. For the structural representations we have followed the

main data structures (e.g., data streams) present in the most used hydrographic data formats (eXtended Triton Format, Generic Sensor Format, Kongsberg EM series, etc.) as well as the work done for XDR and BinX (variable and fixed length arrays, simple structures, strings, unions, etc.) (Eisler, 2006; Kongsberg Maritime AS, 2013; SAIC, 2012; Triton, 2013; Westhead and Bull, 2003).

Based on the above considerations, the resulting HUDDLer middleware implementation for generating code parsers is partly simplified. From the user point of view, the huge benefit is that there is no requirement to write 'homebrew' data parsers. Any time a new release of a format of interest is available, a HUDDLer user has to simply recover the updated schema from the online repository, and re-run the HUDDLer compiler. In principle, it is also possible to use the HUDDL format schemas directly in a real-time universal translator to inspect data for validity (i.e., without first generating an optimized code library for the format).

The HUDDLer code parsers can be used by data processing applications to access binary data, with the advantage of much wider data accessibility (Figure 9).

In addition, we intend to realize a mechanism to notify interested users about format releases. The evident benefit is that everyone knows as soon as a change is made (and users do not have to search for changes when there is a sudden problem in reading a data format).

Another mainstream benefit coming from adoption of HUDDL is automated creation of standardized documentation through the use of XML stylesheet technology (Figure 9). The expected result is a single source for documentation of code, always up to date and consistent.

Candidate additional features for the HUDDL framework could be:

- A mechanism that, using XML Schemas, allows the definition of output code templates, i.e., user-specified code to handle particular data types which is then customized by HUDDLer to the particular structure of the data being read, and integrated with the rest of the library. This may permit great flexibility to the user reducing the amount of bridge code required).
- Handling compressed structures of data (e.g., GZip and Deflate) in case some hydrographic data formats will introduce some sort of real-time compression.
- Signability, which is having a mechanism that permits creation, storing and validation of digital signatures provided by a manufacturer for a released schema.

HUDDL can support the hydrographic community on at least at three different levels (Figure 10):

- At the descriptive level, the user simply takes advantage of the common format repository as well as the standardized templates for documentation.

- At the raw data level, users can use the automatically created raw data parsers. That is, each parser is tailored for a given data format (with all the implicit data peculiarities) as described in the HUDDL description schemas (as they are compiled by HUDDLer).
- At the abstract data level, an additional layer of homogenization with the main aim of simplifying access to hydrographic data (e.g., the same function *getDepthData()* for obtaining the collected depth from various data formats). This additional layer, called a Hydrographic Data Factory, may also be useful for researchers coming from fields not directly related to the ocean mapping.

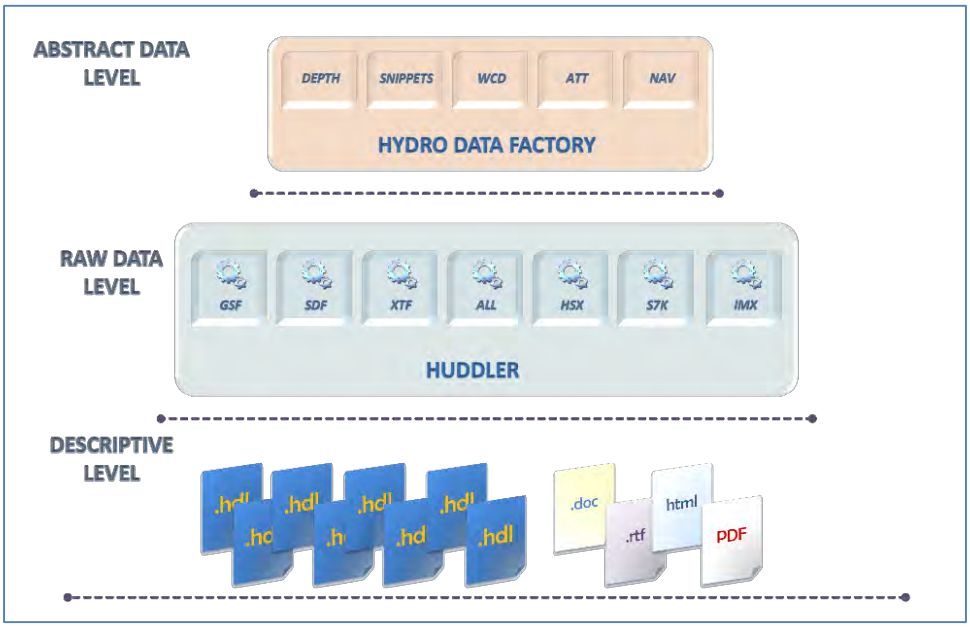

Figure 10 - The three expected levels of users for the HUDDL framework.

Conclusions

Currently, the hydrographic community has to deal with a multiplicity of data formats. Each format was home grown within its specialty and in many cases is based on manufacturer technology. Although new generic data formats have been introduced, there has not been a sufficiently strong reason to rally around one particular method for containing data, nor was any one format general enough to accommodate everyone's needs. The result has been an invasive sea of data formats.

At the same time, the temptation to convert all collected data to a selected data format does not appear to be the optimal solution, since many data formats are fundamentally incompatible with each other, and much metadata and information can be lost during this processing (or even worse, mistranslated).

The intent of HUDDL is not to describe every kind of binary data format that people have ever sent or will ever want to send from machine to machine. Rather, HUDDL just focuses on the

most commonly used hydrographic data formats, so that applications based on its descriptive schemas can be easily written in common computer languages.

Many type of possible applications could benefit from this task-oriented approach: data explorers, conversion tools, metadata archives, etc. Furthermore, once a hydrographic data format/version is correctly described by an XML Schema, the data in that format can be accessed on any platform regardless of the native configuration of the file system.

At the same time, there are also advantages in using a data-description language such as HUDDL versus using diagrams (e.g., UML). HUDDL is more formal than diagrams (leading to less ambiguous descriptions of data formats) and easier to understand (allowing software developers to focus on other issues instead of the low-level details of bit encoding). Also, there is a close analogy between the types used by HUDDL and a high-level language such as C/C++ or Python. This makes the implementation of HUDDL schemas an easy task. Finally, the language specifications itself are XML Schemas that can be passed from machine to machine to perform on-the-fly data interpretation.

Future hydrographic data formats may include the HUDDL Format Schema as part of the binary file itself. This will avoid the risk of breaking the correspondence between the binary data file and the XML Schema description. Another alternative approach to this problem could be to use the first bytes of the file as an integer representing the unique ID reported in a future XML Hydrographic Formats Catalogue.

HUDDL represents a solution for both software and hardware manufacturers to providing a strong and universal mechanism for version control of hydrographic data formats. Developers will have an abstraction tool for development of binary data readers and converters. In addition, the proposed unique Web repository for HUDDL schemas is a powerful solution for spreading the publication of a format update to all the interested hydrographic players (using popular electronic mechanisms as tweets, an RSS or a mailing list). Based on these considerations, we believe that HUDDL represents a way to reduce, in a relatively short time, existing problems related to interoperability and access to hydrographic data.

With increasing (hopefully) amounts of hydrographic data that is publicly available, it will become more and more relevant to make the data accessible and usable by people that are not the collectors, or even necessarily specialists in the area. The value of data increases when all researchers are able to share and interact with each other's knowledge (Georgieva et al., 2009). Since the web is currently one of the main sources to retrieve scientific data, HUDDL schemas coupled with a standard set of metadata represents a solution to understand how the data are stored, and the HUDDL-based code generator represents a further step toward data accessibility and usability. Research success is no longer just a case of capturing and processing better data, it is also the ability to share that knowledge and to provide a way for colleagues to verify scientific outcomes.

References

Calder, B., 2013. Parallel & Distributed Performance of a Depth Estimation Algorithm, U.S. Hydrographic Conference (US HYDRO): New Orleans, LA, USA, p. 11.

Chiu, K., Devadithya, T., Wei, L., Slominski, A., 2005. A binary XML for scientific applications, e-Science and Grid Computing, 2005. First International Conference on, pp. 8 pp.-343.

Eisler, M., 2006. XDR: External data representation standard.

Folk, M., Heber, G., Koziol, Q., Pourmal, E., Robinson, D., 2011. An overview of the HDF5 technology suite and its applications, Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases. ACM: Uppsala, Sweden, pp. 36-47.

Futrelle, J., McGrath, R.E., 2011. Daffodil Parser: Explorations of the DFDL Standard. National Center for Supercomputing Applications, University of Illinois, Urbana--Champaign.

Georgieva, J., Gancheva, V., Goranova, M., 2009. Scientific data formats, In: Mastorakis, N.E., Demiralp, M., Mladenov, V., Bojkovic, Z. (Eds.), 9th WSEAS International Conference on Applied Informatics and Communications (AIC '09). WSEAS Press: Moscow, Russia, pp. 19-24.

Hua, H., Weiss, B., 2011. Strategies for Infusing ISO 19115 Metadata in Earth Science Data Systems, AGU Fall Meeting Abstracts, p. 04.

IEEE, 2008. IEEE Standard for Floating-Point Arithmetic. IEEE Std 754-2008 1-70.

ISO, 2008. ISO/TS 19139-2007 Geographic information -- Metadata -- XML schema implementation. International Organization for Standardization, p. 111.

Kaur, G., Fuad, M.M., 2010. An evaluation of Protocol Buffer, IEEE SoutheastCon 2010 (SoutheastCon), Proceedings of the, pp. 459-462.

Kongsberg Maritime AS, 2013. EM Series Datagram formats - Instruction Manual, p. 126.

Lum, V.Y., Shu, N.C., Housel, B.C., 1976. A General Methodology for Data Conversion and Restructuring. IBM Journal of Research and Development 20(5) 483-497.

Powell, A.W., Beckerle, M.J., Hanson, S.M., 2011. Data Format Description Language (DFDL) v1. 0 Specification. Report of the Open Grid Forum. Retrieved from www. ogf. org/documents/GFD.

Pratt, M.J., 2001. Introduction to ISO 10303—the STEP Standard for Product Data Exchange. Journal of Computing and Information Science in Engineering 1(1) 102-103.

Ramachandran, R., Graves, S., Conover, H., Moe, K., 2004. Earth Science Markup Language (ESML):: a solution for scientific data-application interoperability problem. Computers & Geosciences 30(1) 117-124.

SAIC, 2012. Generic Sensor Format Specification v.03.04. SAIC, p. 151.

Schenck, D., Wilson, P.R., 1994. Information modeling: the EXPRESS way. Oxford University Press New York.

Shu, N.C., Housel, B.C., Taylor, R.W., Ghosh, S.P., Lum, V.Y., 1977. EXPRESS: a data EXtraction, Processing, and Restructuring System. ACM Trans. Database Syst. 2(2) 134-174.

Triton, 2013. eXtended Triton Format (XTF) Rev. 35 Triton Imaging, Inc. , p. 44.

Varda, K., 2008. Protocol Buffers: Google's Data Interchange Format, http://google-opensource.blogspot.com/2008/07/protocol-buffers-googles-data.html.

Westhead, M., Bull, M., 2003. Representing Scientific Data on the Grid with BinX–Binary XML Description Language. EPCC, University of Edinburgh.

Widener, P., Eisenhauer, G., Schwan, K., 2001. Open metadata formats: efficient XML-based communication for high performance computing, High Performance Distributed Computing, 2001. Proceedings. 10th IEEE International Symposium on, pp. 371-380.

Yongguo, J., Lianying, L., Zhongwen, G., 2009. Design of Marine Information Metadata and Directory Service System Based on XML, Database Technology and Applications, 2009 First International Workshop on, pp. 574-577.

Author biographies

Giuseppe Masetti received a MS degree in Ocean Engineering (UNH, USA) in 2012, and a Master in Marine Geomatics (2008) and a Ph.D. degree (2013) in System Monitoring and Environmental Risk Management (University of Genoa, Italy). At CCOM/JHC he works mainly on signal processing for marine target detection.

Brian Calder is an Associate Research Professor and Associate Director at CCOM (UNH, USA). He has a Ph.D. in Electrical and Electronic Engineering, completing his thesis on Bayesian methods in SSS processing (1997). He is currently focusing on statistically robust automated data processing approaches and tracing uncertainty in hydrographic data.