Canadian Investigations of the Keeley Bricks With Application to Profile Data Transfer Using XML

Anthony W. Isenor
Defence R&D Canada – Atlantic
anthony.isenor@drdc-rddc.gc.ca

Introduction

The 2002 SGXML meeting in Helsinki, Finland, outlined various action items for the Group. In particular, Action Items 5 and 7 under subsection 6.2 [1] identified a specific point data investigation. This report briefly describes the development resulting from this investigation.

The development involved three Canadian data centres: the Institute of Ocean Sciences (Sidney, B.C.), the Marine Environmental Data Service (Ottawa, Ontario) and the Bedford Institute of Oceanography (Halifax, N.S.). A detailed report of the development is available [2, 3]. Here, a summary is presented and details of the data structure are described.

Background

The 2002 Helsinki meeting introduced the SGXML to the concept of generic data objects for packaging ocean data. These objects were called Keeley Bricks. The original concepts behind the bricks were based on the work of Bob Keeley, Marine Environmental Data Service, Canada.

The Keeley Bricks are essentially data objects that group associated data and information. The term “bricks” is used because these objects resemble construction bricks in that they can be assembled in different ways to produce different structures. This represents the two fundamental principles behind the Keeley Brick concept: 1) exploit the natural grouping of data and information into well-defined objects, and 2) arrangement of the bricks in various ways to imitate the natural structures found in a variety of ocean data types.

A Canadian investigation has constructed the initial set of bricks required to define a structure appropriate for ocean profile data. This work included full brick definitions. The work also extended into an application of the bricks in an XML environment. This resulted in brick definitions that focused on the associated XML syntax. Thus, the brick definitions have moved beyond the abstract, to specific implementation details related to elements and attributes in an XML environment.

The bricks resulting from the Canadian development are summarised in Table 1. Both pure and compound bricks are identified. Compound bricks are specific to the XML application. Compound bricks are wrappers around other compound or pure bricks. Pure bricks contain data and information content. In the XML application, this content is stored in the elements and attributes.

The profile data structure resulting from the development is shown in Figure 1. Here, compound bricks are identified in green (also identified by the trailing string "_set") while pure bricks are red. The figure also shows the XML occurrence of the brick. The square brackets enclose the minimum and maximum occurrence of the brick using the notation [i, j] with i representing the minimum and j the maximum.

An example of encapsulating water sample data in the XML structure is provided in Annex 1. Only one record of data is shown. The data record contains five data values.

Detailed information on the bricks and the profile data structure is available [3]. In this brief report, some of the specific implementation decisions are described. The reader is assumed to have an intermediate knowledge of XML [4, 5, 6].

Parameter Codes

Parameter codes represent the coding of information related to the variables in a dataset. Codes are often stored in systems called parameter dictionaries. Within the XML profile data structure, the dictionary can be described by name using the data_dictionary brick. This brick is contained in the data_collection compound brick at the highest level of the profile structure (see Figure 1).

At the individual XML datum level, the specification of a parameter code may occur in one of three forms:

1) the code as a tag name,
2) the code as tag content, and
3) the code as attribute content.

The parameter code existing as a tag name was not considered viable. In essence, this option would allow for many thousands of tags, as there are many thousands of oceanographic parameters. This option would also eliminate the usefulness of the schema validation process. This is because one would not be able to create a schema that identified all possible combinations of parameters.

Considering option 2), the parameter code could easily exist as content for the tag. However, if the data value is also stored in tag content, then the direct connection (or

Table 1. The list of bricks and definitions. Pure and Compound bricks are described in the text.

BRICKS	BRICK TYPE	DEFINITIONS
analysis_method	Pure	Stored information about any physical, chemical or biological analyses carried out on the data
availability	Pure	Stores information about the possible release of the data to the public
calibration	Pure	Stores calibration information on the instrument, sensor or variable
comment	Pure	Stores general textual information not intended to be used in data retrievals
data_collection	Compound	Used to encapsulate the entire XML file
data_dictionary	Pure	Indicates the specifics of the data dictionary being used within the collection
data_point	Pure	Used to store any type of data or metadata value
data_set	Compound	Used to encapsulate a dataset at a defined level of granularity
data_set_id	Pure	A numeric or text identifier for a particular data set
depth_pressure	Pure	Store the z coordinate of the data
history	Pure	Processing history of the data
history_set	Compound	Used to encapsulate history information
instrument	Pure	Information about the instrument used to make the measurements.
latitude	Pure	The y coordinate of the data
ldate	Pure	The time coordinate of the data
location_set	Compound	Used to encapsulate the x, y, z, t values.
longitude	Pure	The x coordinate of the data
previous_value	Pure	Information about the value before it is changed
provenance	Pure	The originator of the data
quality	Pure	A marker providing an assessment of data quality
quality_testing	Pure	Information about how the data quality assessment was made
sampling	Pure	Information about the sampling methods used
sensor	Pure	Identifies sensor specifics
units	Pure	The units of measurement
variable	Pure	Information about the variables measured.
variable_set	Compound	Used to encapsulate all the information required to declare a variable

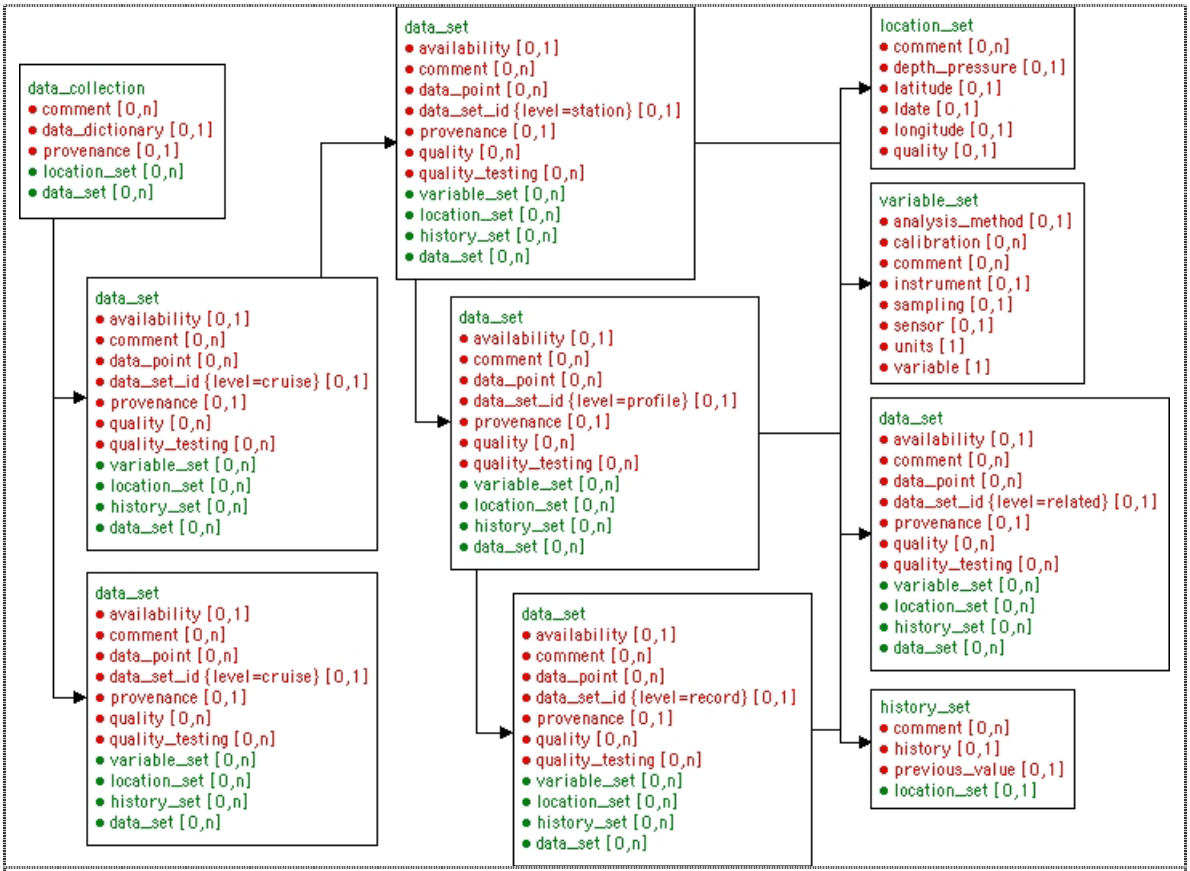

Figure 1. The profile data structure using Keeley Bricks. The red text indicates pure bricks, while the green text indicates compound bricks. Compound bricks may also be identified by the name, which typically ends in the string "_set" (the exception being data_collection, which is also a compound brick). The [i, j] notation indicates the [minimum, maximum] occurrence of the brick. The arrows indicate an expansion of one element into subelements in the XML application. Components of the structure are described in the text. The details and definitions of the bricks may be found at [3].

encapsulation) between the code and the data value is lost. This connection could exist if the attribute of the tag contained the data value.

Option 3) was selected for this project. This option packages the code with the data value in a single XML element. As well, the option leaves open the possibility of using a list of allowable parameters for the attribute content.

The pt_code attribute (see Annex 1, data_point tag) is used to store parameter code information in various bricks. As well, a pt_link attribute may be used to provide a counter on the pt_code. The pt_link attribute allows for the same parameter occurring more than once in a dataset, as might occur if oxygen values are determined using two different methods.

The use of the schema in document validation is an important functionality within XML. The schema defines allowable structures for the XML documents. The XML environment provides tools that compare and report on the compliance of a particular XML document as compared to the schema. Content rules may be built into the schema, thereby reducing the requirement on developed software. An example content rule may be a range for latitude between –90 and +90 degrees.

data_set Compound Brick

The data_set compound brick (see Figure 1) is an important construct within the profile structure. It was recognized that the term "dataset" means a variety of things to different people. With this in mind, the data_set compound brick contains an identifier brick, data_set_id, which identifies the particular level of granularity of the data_set.

The data_set compound brick contains the necessary compound and pure bricks to describe:

- the availability of the dataset for distribution,
- free-format comments,
- data points within the dataset,
- a dataset identifier,
- the owner of the dataset,
- the quality tests applied and the results of those tests,
- the variables contained within the dataset,
- spatial-temporal location information for the dataset,
- the processing history of the dataset, and
- a dataset at a finer level of granularity.

A dataset within a dataset exploits the natural hierarchy familiar to oceanographic data collections (e.g., many profiles at a station, and many stations within a cruise). The last bullet in the above list indicates this concept.

There are four bricks of particular importance within the data_set compound brick: data_point, data_set_id, variable_set and location_set. These will now be described in more detail.

The data_point brick is used to encapsulate data or metadata relevant to the particular data_set level. The content of data_point is the actual data or metadata. data_point has

four attributes (see Figure 2) which contain the parameter code, a point link, statistical properties and the type of content. Type can be numeric, character, date, time or date/time. The statistic attribute identifies if the data_point is a statistical measurement such as a mean, standard deviation, etc.

data_point {**pt_code**, pt_link, statistic, typing}

Figure 2. The data_point brick. The content for data_point is any data or metadata value for the dataset. Attributes of the brick are shown inside {}. The bold text indicates mandatory content.

The data_set_id brick (Figure 3) allows for identification of the particular dataset. The content of this brick is a unique dataset identifier. The level attribute for the brick identifies the granularity of the dataset. For example, allowable content in the level attribute includes cruise, station, profile, and record. This content was developed for the profile structure. Other content may be required for other data types.

data_set_id {**level**}

Figure 3. The data_set_id brick. The content of data_set_id is a unique identifier for the dataset. Attributes of the brick are shown inside {}. The bold text indicates mandatory content.

The variable_set compound brick contains all the information to describe a variable or parameter used within the XML document. This compound brick (Figure 4) is comprised of bricks that allow for descriptions of:

- analysis methods used on particular variable,
- calibration information for the variable,
- free-format comments,
- information on the instrument used to collect the variable,
- information on the sampling used to collect the variable,
- information on sensors used to collect the variable,
- units of the variable, and
- variable specific information such as name, limits, accuracy and precision.

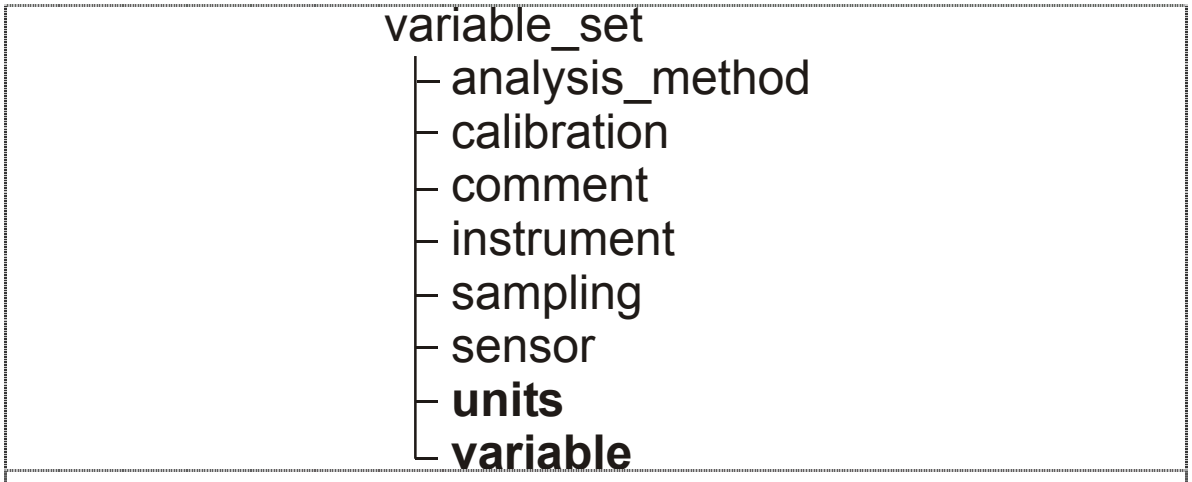

Figure 4. The variable_set compound brick. The vertical, branching line, illustrates the brick structure encapsulated within variable_set. This compound brick contains eight pure bricks. The content of the eight bricks is not shown. The bold text indicates mandatory bricks.

The units brick is of particular importance (Figure 5) within the variable_set compound brick. The units brick contains all the information related to the units of the variable. The pt_code and pt_link attributes are contained within the units brick to allow usage outside the variable_set compound brick. In the profile implementation, the units and variable bricks are both contained within variable_set. This arrangement does not require pt_code and pt_link in the units brick, as the encapsulation within variable_set implies a relationship to the variable brick. However, in anticipation of the units brick being used outside the variable_set compound brick, we include the pt_code and pt_link attributes.

The units brick also has attributes received_units and stored_units. received _units is used to contain the units of the variable as originally received by the agency constructing the XML document. The stored_units attribute contains those units used within the particular XML document.

The units brick also contains subelements conversion, reference and variable_name. The variable_name element is similar to pt_code and pt_link in that it is not necessary within the profile implementation. Again, it is included for anticipated use of the units brick outside the variable_set compound brick. The conversion element allows a description of the conversion from the received to the stored units. Finally, the reference element allows for a publication reference for those detailed conversions.

Figure 5. The units brick. This brick contains 4 attributes and three elements. The bold text indicates mandatory content.

The location_set compound brick (see Figure 6) contains the position and time specification for the dataset and quality information for the spatial-temporal point. In particular, the main bricks within the location_set compound brick are latitude, longitude, depth_pressure and ldate (a location date).

The content bricks within location_set represent a slight departure from the general philosophy of the Keeley Bricks. The content bricks have very specific tag names. This specific naming was intended to highlight the importance of these parameters.

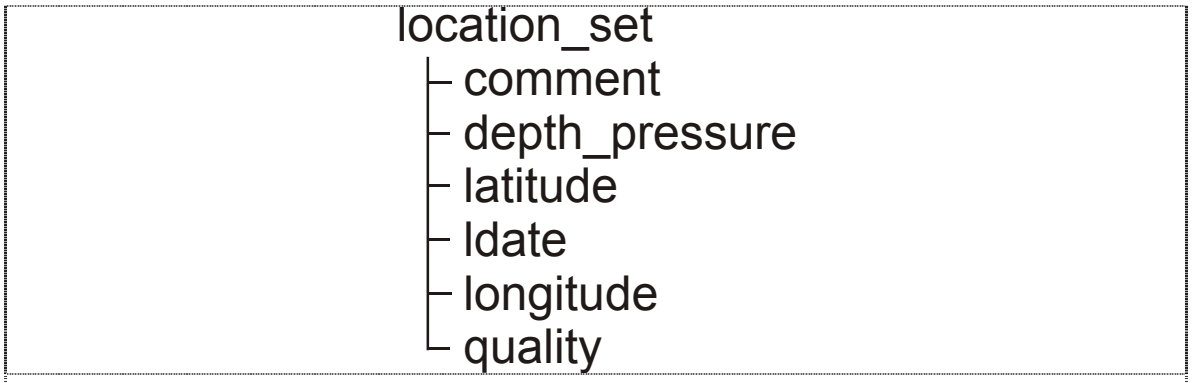

Figure 6. The location_set compound brick. The lack of bold text indicates there is no mandatory content.

Unfortunately, the specific naming of these bricks caused numerous problems and represents one of those areas where improvement could be made to the profile structure and quite possibly the bricks in general. The problems resulted from the added complexity at the record level, when these specifically named parameters are data (see section on Interesting Results).

Conversion Methods

Various XML documents were produced using the developed XML profile structure. All three participating labs provided an assortment of profiling float, XBT, water sample, and CTD data in the XML structure. This required the development of conversion software at two levels: the conversion from in-house to XML structure and the conversion from XML structure to in-house.

The method used for the first conversion is specific to the in-house development environment, and therefore of limited use to the international community. The second conversion method, from XML to in-house format, is more interesting for the community. This conversion was conducted in Fortran using msxml (IOS and MEDS), and in Java using data binding techniques (BIO). As well, eXtensible StyleSheet Language Transformations (XSLT) were used as a preliminary investigative tool with limited success (BIO).

At BIO, the transformation of ODF profile data to XML was conducted using the Ocean Sciences Division (OSD) Matlab-based set of tools called the Oceans Data System (ODS) Toolbox. This suite of tools has been developed by OSD over the past decade and constitutes the primary analysis tool within the Division. The Toolbox is capable of reading ODF files and creating a memory resident ODF cell structure within the Matlab environment.

The BIO transformation from XML to ODF was conducted using two different methods. The first method used XSLT, an XML-based language used to manipulate the structure of an XML document. This method was developed as an exercise in understanding the usefulness and versatility of XSLT and was not considered the most practical method of dealing with the brick objects.

The second BIO method for transforming the XML to ODF utilized Java technology – in particular, data binding using Sun's Java Architecture for XML Binding (JAXB). Data Binding is a technique for linking and automatically creating Java classes based on a constraint file, such as a schema file. The result of the binding is the removal of complicated data access methods via calls to nodes in the XML structure. Instead, the binding creates more descriptive classes based on the brick names. So, classes such as DataCollection were created. As well, binding results in method calls such as object.getProvenance().getDescription() (where object might be DataCollection) to obtain the description data within the provenance brick.

The JAXB-developed classes are really object representations of the bricks. A similar construction of objects may use an object-oriented database. In such a database, the structure, including encapsulation and inheritance in the XML document, can be utilized within the database.

After the classes and methods were created in the binding process, a wrapper program, called ProcessDataCollection, was created. ProcessDataCollection controlled the transfer (unmarshalling) of the XML document into Java objects. ProcessDataCollection also controlled the access of the Java objects to create a new file in the in-house ODF format.

Some Interesting Results

We conclude with several interesting results discovered during the development process.

- During the development, some tags for data or metadata were given specific names (e.g., latitude, longitude, depth_pressure, ldate). Such specific naming is not necessary within the XML environment. These data could be included using the data_point brick and the pt_code attribute.

  This specific naming complicated the BIO XSLT investigation. The latitude, longitude and ldate bricks were sufficiently specific to remove the requirement for the pt_code attribute within these bricks. Although seemingly innocuous, the removal of the pt_code attribute caused problems with the XSLT code for transforming the XML to ODF. Specifically, the pt_code (and pt_link) attribute permitted the sorting of the elements. This was very important for the output ODF structure, as ODF requires the header variable definitions in the ODF file to be in the same order as the variable columns. Using a sort on pt_code for both variable and data_point information resulted in a consistent ordering of the variable information in the ODF file. The removal of the pt_code attribute in the latitude, longitude and depth_pressure bricks meant special consideration was required for these data. This introduced complications in the XSLT code.

- Another realization during the process dealt with the numerous inadequacies with the in-house formats. These formats were developed to maintain local processing and archival systems. The data model on which many were developed was not oriented towards ocean data in a general sense, but more toward specific ocean data types. For example, the ODF format does not have the ability to store information on more than one instrument. This implies that the data in a single ODF file must originate from a single instrument (not always the case). The XML structure, allows unique data types to identify unique instruments. This more generalized structure does not map well to the ODF format.

- The development presented here only provides a data structure for encapsulating ocean data. When exchanging data within a common structure, the problems associated with parameter codes are still present. For example, one institute may refer to particular organic carbon as CPX1 in milligrams/litre while another institute refers to the same data as Carbon:Particulate:Organic in

micrograms/litre. Although the identification of the parameter dictionary used within the XML document may be included using the data_dictionary brick, the conversion from one dictionary to another remains a requirement. Developments are ongoing to convert code sets from one dictionary to another using XML [7].

- Null values are not XML friendly. In the old paradigm of data formats, null values were defined to fill the space of the "missing" data value. These null values were typically outside the space of realizable data values. For example, latitude stored in degrees might have an associated null value of –99. The –99 is not in the realizable range of latitude values expressed in degrees. In XML, one may define restrictions to set the allowable limits of data values. If such a schema restriction were placed on latitude for the range –90 to 90, the null value of –99 would not pass validation.

  Also, implementing this restriction would not allow an empty tag to be present in the XML document. In XML, all tag content is used in validation, including empty content.

  This functionality has consequences for the mandatory set of tags defined for the bricks. If, in the schema, a tag is declared as **mandatory** with **restrictions**, then the tag **must** be present with valid content in the XML document. The input data streams therefore must have that content available.

- The XML documents generated within this project provided an opportunity to investigate file size issues. The ODF files, which are ASCII, were compared to the XML documents generated from the ODF content. When comparing six ODF files (bottle, CTD, XBT, float, moored current meter, and underway TS data) to the XML equivalents, the XML representation occupied 600% of the disk space as compared to the ODF files. However, when compressed using common compression software, the XML representation was 40% larger as compared to the compressed ODF files. This indicates that compression of XML files may alleviate XML file size issues.

The Future

This project has shown that XML is a viable exchange mechanism for ocean profile data. The project has also shown that development based on the Keeley Bricks shows promise as an object-oriented approach for developing XML structures for ocean data types. Building on the work of SGXML, the development of code mapping capabilities [7] has also shown that seamless exchange of ocean data can be a reality.

The future of international ocean data exchange will ultimately rest with institute personnel responsible for developing the exchange systems. At this time, the technical problems of exchange are minimal. Potentially the largest problem is developing the interest and initiative to see a vision to completion. The vision may start small, with the linking of a few centres all providing data in a common structure and parameter dictionary understood by the client. It could easily grow to a single portal interface that provides the client with access to data from the international data system. The portal could interrogate local databases and provide datasets to clients in a common structure and dictionary. This has been described [8] as the Mikhailov Model of a distributed ocean data system. It can be a reality.

References

1. Report of the ICES-IOC Study Group on the Development of Marine Data Exchange Systems Using XML, Helsinki, Finland, 15–16 April 2002, ICES CM 2002/C:12.

2. Isenor, Anthony W., J. Robert Keeley and Joe Linguanti. Developing an eXtensible Marup Language (XML) Application for DFO Marine Data Exchange via the Web, Defence R&D Canada – Atlantic ECR 2003-025, March 2003.

3. Canadian XML Brick Website, http://www.meds-sdmm.dfo-mpo.gc.ca/meds/Prog_Int/ICES/ICES_e.htm

4. For general knowledge see http://www.oasis-open.org/cover/xml.html#applications

5. For Specifications see http://www.w3.org/

6. Means, W. Scott and Elliotte Rusty Harold. XML in a Nutshell, January 2001, 0-596-00058-8.

7. Isenor, Anthony W. 2003. XML Based Manipulation of Codes Exchanged Between Data Systems, Technical Memorandum in preparation, Defence R&D Canada – Atlantic.

8. Report of the Working Group on Marine Data Management, Helsinki, Finland, 17–19 April 2002, ICES CM 2002/C:11.

Annex 1 – Example XML Document

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<data_collection xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="c:\Anthony\Projects\xml\odf_conversion\brick
s_v2.xsd">
   <data_dictionary>
      <dictionary_name>GF3+</dictionary_name>
   </data_dictionary>
   <data_set>
      <data_set_id level="cruise">73025</data_set_id>
      <provenance>
         <agency>Bedford Institute</agency>
         <date_created>2003-02-18Z</date_created>
         <institute_code>1810</institute_code>
         <originator_identifier>73025</originator_identifier>
      </provenance>
      <data_set>
         <data_set_id level="station">118</data_set_id>
         <data_set>
            <comment>17-JAN-1984 00:00 BOTTLE DATA 73027</comment>
            <comment>17-JAN-1984 00:00 </comment>
            <data_set_id level="profile">Q1</data_set_id>
            <provenance>
               <agency>Bedford Institute</agency>
               <data_grouping>BOTL</data_grouping>
               <date_created>2003-02-18Z</date_created>
               <description>Q2</description>
               <originator_identifier>Q1</originator_identifier>
            </provenance>
            <variable_set>
               <instrument type="bottle"/>
               <sampling pt_code="PRES">
                  <interval>   0.00000</interval>
               </sampling>
               <units pt_code="PRES" stored_units="decibars"/>
               <variable kind="I" pt_code="PRES" typing="R">
                  <decimal_places>4.000000</decimal_places>
                  <maximum_value>411</maximum_value>
                  <minimum_value>33</minimum_value>
                  <null_value>-.99000000D+02</null_value>
                  <variable_name>PRES_1</variable_name>
               </variable>
            </variable_set>
            <variable_set>
               <instrument type="bottle"/>
               <units pt_code="PSAL" stored_units=""/>
               <variable kind="D" pt_code="PSAL" typing="R">
                  <decimal_places>4.000000</decimal_places>
                  <maximum_value>34.944</maximum_value>
                  <minimum_value>32.725</minimum_value>
                  <null_value>-.99000000D+02</null_value>
                  <variable_name>PSAL_1</variable_name>
               </variable>
            </variable_set>
            <variable_set>
```

```
   <instrument type="bottle"/>
   <units pt_code="DOXY" stored_units="ml/l"/>
   <variable kind="D" pt_code="DOXY" typing="R">
      <decimal_places>4.000000</decimal_places>
      <maximum_value>8.53</maximum_value>
      <minimum_value>7.36</minimum_value>
      <null_value>-.99000000D+02</null_value>
      <variable_name>DOXY_1</variable_name>
   </variable>
</variable_set>
<variable_set>
   <instrument type="bottle"/>
   <units pt_code="SLCA" stored_units="mmol/m**3"/>
   <variable kind="D" pt_code="SLCA" typing="R">
      <decimal_places>4.000000</decimal_places>
      <maximum_value>0.8</maximum_value>
      <minimum_value>0.6</minimum_value>
      <null_value>-.99000000D+02</null_value>
      <variable_name>SLCA_1</variable_name>
   </variable>
</variable_set>
<variable_set>
   <instrument type="bottle"/>
   <units pt_code="PHOS" stored_units="mmol/m**3"/>
   <variable kind="D" pt_code="PHOS" typing="R">
      <decimal_places>4.000000</decimal_places>
      <maximum_value>7</maximum_value>
      <minimum_value>1</minimum_value>
      <null_value>-.99000000D+02</null_value>
      <variable_name>PHOS_1</variable_name>
   </variable>
</variable_set>
<variable_set>
   <instrument type="bottle"/>
   <units pt_code="TEMP" stored_units="degrees C"/>
   <variable kind="D" pt_code="TEMP" typing="R">
      <decimal_places>4.000000</decimal_places>
      <maximum_value>0.9</maximum_value>
      <minimum_value>-1.75</minimum_value>
      <null_value>-.99000000D+02</null_value>
      <variable_name>TEMP_1</variable_name>
   </variable>
</variable_set>
<location_set>
   <ldate property="creation">
      <pdate>2003-02-18Z</pdate>
      <ptime>19:30:53.99Z</ptime>
   </ldate>
</location_set>
<location_set>
   <ldate property="original">
      <pdate>1990-11-20Z</pdate>
      <ptime>00:00:00Z</ptime>
```

```
   </ldate>
</location_set>
<location_set>
   <latitude property="start">  67.00700</latitude>
   <ldate property="start">
      <pdate>1973-09-04Z</pdate>
      <ptime>00:00:00Z</ptime>
   </ldate>
   <longitude property="start"> -26.83800</longitude>
</location_set>
<location_set>
   <latitude property="end">  67.00700</latitude>
   <longitude property="end"> -26.83800</longitude>
</location_set>
<history_set>
   <comment> </comment>
   <history>
      <application_date>1992-11-05Z</application_date>
   </history>
</history_set>
<history_set>
   <comment>GF3 Name Checking and Code Formatting</comment>
   <comment>Name check performed</comment>
   <history>
      <application_date>2003-02-18Z</application_date>
   </history>
</history_set>
<data_set>
   <data_point pt_code="min_depth">  32.66939</data_point>
   <data_point pt_code="max_depth"> 406.51097</data_point>
   <data_point pt_code="sounding"> 424.00000</data_point>
   <data_point pt_code="depth_off_bottom"> 0.00000</data_point>
   <data_set_id level="related"/>
   <variable_set>
      <instrument type="sounder"/>
      <units pt_code="sounding" stored_units="metres"/>
      <variable kind="I" pt_code="sounding" typing="R">
         <decimal_places>5.000000</decimal_places>
         <variable_name>sounding</variable_name>
      </variable>
   </variable_set>
   <variable_set>
      <instrument type="bottle"/>
      <units pt_code="max_depth" stored_units="metres"/>
      <variable kind="I" pt_code="max_depth" typing="R">
         <decimal_places>5.000000</decimal_places>
         <variable_name>maximum depth</variable_name>
      </variable>
   </variable_set>
   <variable_set>
      <instrument type="bottle"/>
      <units pt_code="min_depth" stored_units="metres"/>
      <variable kind="I" pt_code="min_depth" typing="R">
```

```
                     <decimal_places>5.000000</decimal_places>
                     <variable_name>minimum depth</variable_name>
                  </variable>
               </variable_set>
               <variable_set>
                  <instrument type="sounder"/>
                  <units pt_code="depth_off_bottom" stored_units="metres"/>
                  <variable kind="I" pt_code="depth_off_bottom" typing="R">
                     <decimal_places>5.000000</decimal_places>
                     <variable_name>depth off bottom</variable_name>
                  </variable>
               </variable_set>
            </data_set>
            <data_set>
               <data_point pt_code="PSAL">32.7250</data_point>
               <data_point pt_code="DOXY">8.5300</data_point>
               <data_point pt_code="SLCA">0.6000</data_point>
               <data_point pt_code="PHOS">5.0000</data_point>
               <data_point pt_code="TEMP">-0.5000</data_point>
               <data_set_id level="record"/>
               <location_set>
                  <depth_pressure pt_code="PRES">33.0000</depth_pressure>
               </location_set>
            </data_set>
         </data_set>
      </data_set>
   </data_set>
</data_collection>
```