

Development and Use of Marine XML within the Australian Oceanographic Data Centre to Encapsulate Marine Data

Belinda Ronai, Paul Sliogeris, Matthew de Plater, Krystyna Jankowska
Australian Oceanographic Data Centre.
Maritime Headquarters, Wylde St, Potts Point, NSW, 2011, Australia.
<http://www.aodc.gov.au>

Abstract

The Australian Oceanographic Data Centre (AODC) archives, quality controls and disseminates a large variety of marine data. The extensible Markup Language (XML) has been tailored to encode marine data and provides a clearly defined way to structure, describe and interchange marine data and has been established as the internal data format standard for the AODC. The development, advantages, disadvantages and uses of Marine XML within the AODC have been outlined.

Table Of Contents

Abstract	1
1 Introduction.....	1
2 Design Goals	2
3 Representing Marine Data.....	2
3.1 Encoding Spatial Data	3
3.2 Encoding Temporal Data	5
3.3 Encoding Marine Data	7
3.4 Control Vocabularies	11
3.5 Auditing.....	11
3.6 Additional Features.....	12
4 Current Status	12
5 Conclusion.....	13
6 Comments	13
References.....	13
APPENDIX A Marine XML Document Type Definition (DTD)	14
APPENDIX B Marine XML Schema	17
APPENDIX C Example Marine XML File.....	25

List Of Figures

Figure 1 Marine XML Structure	3
Figure 2 SpatialReference Element Structure	4
Figure 3 Coordinates Element Structure	4
Figure 4 Defining a Spatial Polygon	5
Figure 5 GeoPolygon Element Structure.....	5
Figure 6 TemporalReference Element Structure.....	5
Figure 7 Period Element Structure	6
Figure 8 Data Element Structure.....	8
Figure 9 Parameter Element Structure	8
Figure 10 ParameterSet Element Structure	9
Figure 11 Edits Element Structure	11

1 Introduction

The Australian Oceanographic Data Centre (AODC) quality controls, archives and disseminates a large variety of marine data. Efficient and effective management of this data are key objectives of the AODC. Advances in technology have created new possibilities in managing data. Markup languages are used for managing all types of data, including spatial data (Geography Markup Language, GML). Markup languages are especially useful where it is intended that data is to be maintained over the long term, as is the case with most oceanographic data centres. Using the extensible Markup Language (XML) to encode marine data provides a clearly defined way to structure, describe and interchange marine data and has been established as the internal data format standard for the AODC.

There are a number of reasons why XML was chosen to encode marine data.

- **Self Describing**
XML is easily readable by both humans and computers. XML defines a set of rules that make interpretation by computer very simple. XML documents remain text-based and self describing, thus negating the need for complex format descriptions.
- **Whole-of-life data management**
The ability of XML to encapsulate data allows the data manager to preserve all raw data, quality flags and edited values in the one data file. This simplifies data management, because the creation of multiple files based on quality is avoided, enabling data to easily be rolled back to any stage if required.
- **Object-Oriented**
Marine data does not fit well into relational data models because of its wide ranging parameters and spatiotemporal attributes. By applying an object-oriented data structure to marine data like XML, data is more intuitively understandable and more naturally organised.
- **Reformatting**
Through the use of extensible Stylesheet Language Transformation (XSLT), data encoded with XML can be transformed and presented in any required format, including HTML pages, Scalable Vector Graphics (SVG) and even PDF files.
- **Metadata**
The data's metadata can be encapsulated within the XML document together with any other information pertinent to the data file. This allows for automatic metadata record generation using XSLT.
- **Data Exchange**
Because of the many reasons listed above, encoding data with XML creates a file that is easily transportable to other organisations, into other formats, to analysis software, into models or into databases.

The Marine XML structure presented in this paper is primarily for internal use within the AODC. The aim of this paper is to present to the wider marine community our approach in developing an XML structure to encode marine data and how it has been implemented within our organisations data management procedures.

2 Design Goals

Marine XML was developed for inhouse use by the AODC with the following design goals.

- To provide a means of encoding all types of marine data for storage and portability.
- To establish a basis for developing a streamlined data management system within the AODC by creating interoperable software, based upon Marine XML encoded data, utilising XML technologies.
- To create easy to understand, self describing encoding of marine data.
- To develop a means to track all quality control processes and edits and operators within the one XML file.
- To reformat data using XSLT into other marine formats to meet national and international data exchange obligations.
- To display data using XSLT straight from XML without reformatting data into graphics files.
- To integrate metadata within the marine data itself.

3 Representing Marine Data

Creating a suitable data format to encapsulate all of the types and forms of marine data is a huge task. The data structure needs to be able to handle varying temporal and geographical extents. It should be possible to specify temporal data as an instant in time or a period (continuous or in-continuous). Geographical extents will need to be specified as a single point, line, box, or polygon region. The structure will need to handle both multi-dimensional and multi-parameter data. It should also have facilities to track any changes on the data, provide details on quality control processes and flag systems used, along with the quality of the data itself, custodian details and any other properties that may need to be described.

Figure 1 illustrates the structure of the Marine XML Schema that has been developed by the AODC to encapsulate marine data. The *MarineDataSet* element is the parent (or root) element of the entire data set. Child elements of *MarineDataSet* can be used to describe the quality of the data, custodian,

quality control and any other various properties pertaining to the data set. These are followed by a series of *MarineDataRecord* elements that contain the data itself. Each individual *MarineDataRecord* element is used to encapsulate a group of data that has the same spatial and temporal extent. The quality of the record can also be specified, along with any number of other properties pertaining to the record.

The remainder of this section breaks down the Marine XML structure into components. Not all elements and attributes are described in detail. The purpose of this paper was to focus on encoding the spatial and temporal extents and the marine parameters. Please see the Marine XML DTD and SCHEMA in Appendix A and B respectively, for the Marine XML specifications in detail.

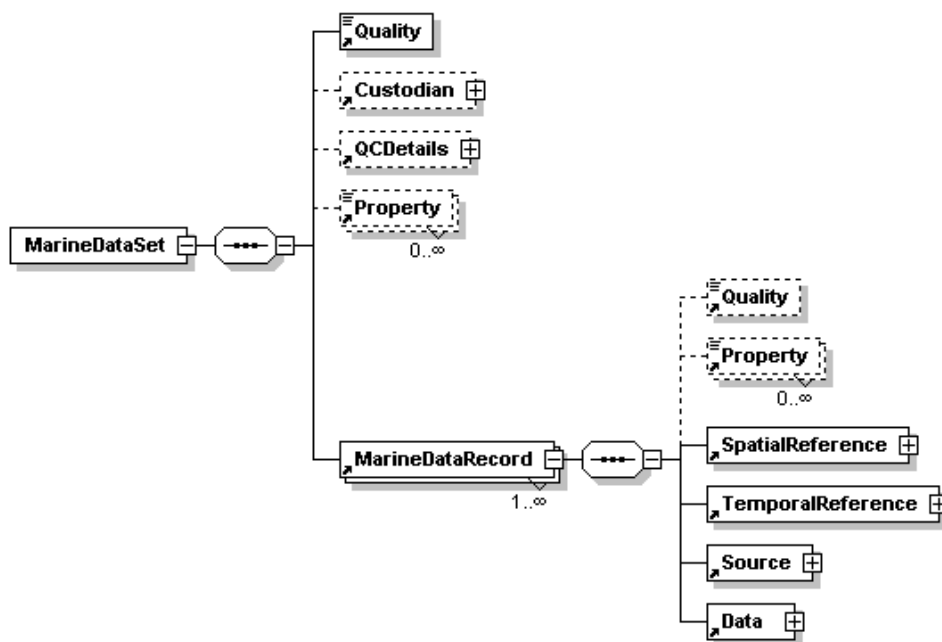


Figure 1 Marine XML Structure

3.1 Encoding Spatial Data

The spatial extent of a marine data observation can take a variety of forms. It can be a single point in space, a line, a box or a polygon region. Although the geographical extent of the majority of observed marine data are single points in space, it is necessary to build the structure of the XML so that it is flexible enough to encode any spatial extent.

Some examples of marine data with varying geographical extents include;

- Single Point Data would be a sea surface temperature measurement,
- Line Data would include data from towed fish such as side scan sonar,
- Polygon Data includes climatological data for a square degree of latitude/longitude at a specified depth,

- Box or 3D Polygon Data might be climatological data for a square degree of latitude/longitude between standard depths or an entire sea or ocean.

The *SpatialReference* element (Figure 2) is used to encapsulate the spatial extent of a *MarineDataRecord*. It may contain a *Quality* child element used to specify the quality of the spatial reference. It should also contain one of the child elements, *GeoPoint*, *GeoBox*, *GeoLine* or *GeoPolygon*.

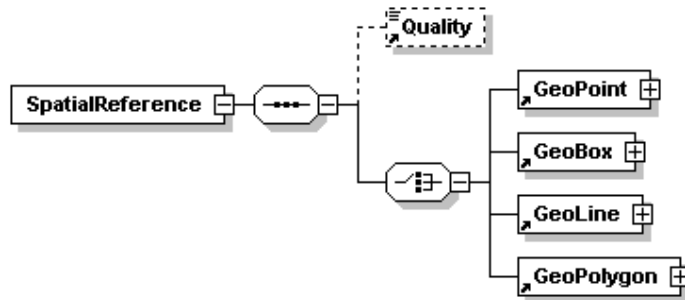


Figure 2 SpatialReference Element Structure

Each of the child elements, *GeoPoint*, *GeoBox* and *GeoLine*, are based on at least one *Coordinates* child element. The *GeoPoint* element has one *Coordinates* that specifies the coordinates of the point in space. The *GeoBox* element must contain two *Coordinates* that specify the opposite corners of the spatial box (minimum values and maximum values). The *GeoLine* element must contain two or more *Coordinates* to specify the spatial line.

The *Coordinates* element (Figure 3) is used to define the coordinates of a single point in space. It contains compulsory child elements *Latitude* and *Longitude* for describing spatial data in a marine context with units being decimal degrees for the greatest accuracy. There is also an option to have a *Depth* or *Altitude* child element if required with units being metres. Both *Depth* and *Altitude* elements were included to avoid uncertainty in regards to meaning of negative values.

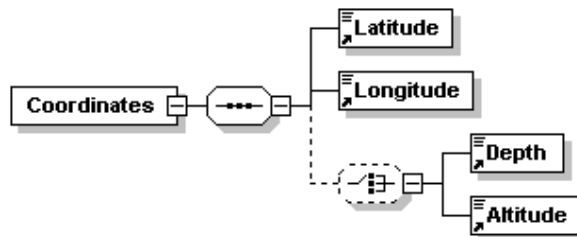


Figure 3 Coordinates Element Structure

The *GeoPolygon* element is a little more complex. It is used to specify a spatial extent in the shape of a polygon. This consists of an *OuterBoundaryIs* and any number of *InnerBoundaryIs* elements (Figure 4 and Figure 5), all of which are specified by a *LinearRing* child element.

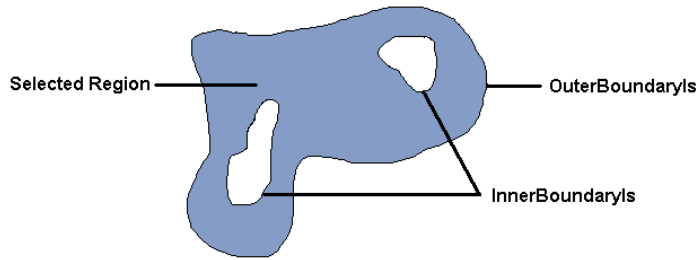


Figure 4 Defining a Spatial Polygon

The *LinearRing* element consists of four or more *Coordinates* child elements to define the linear ring.

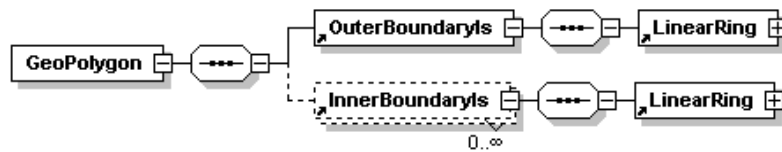


Figure 5 GeoPolygon Element Structure

3.2 Encoding Temporal Data

Temporal data can be expressed either as an instant in time or as a period. The period may either be continuous or periodical (non-continuous). Some examples of marine data with varying temporal extents include;

- Instant such as an observation of sea surface temperature,
- Continuous Period such as data obtained from an Acoustic Doppler Current Profiler (ADCP) or ARGOS Float,
- A Periodical period would be one describing marine climatological data (eg for the month of March).

The *TemporalReference* element (Figure 6) is used to specify the temporal extent of a *MarineDataRecord*. It may contain a *Quality* element to specify the quality of the temporal information. It then may contain either an *Instant*, to specify an instant in time, or a *Period* child element to specify a period in time.

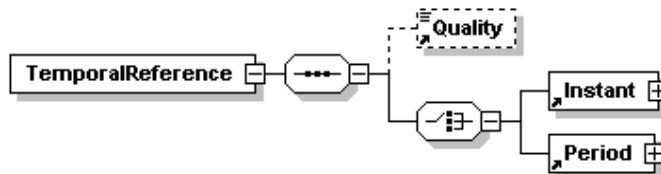


Figure 6 TemporalReference Element Structure

The *Instant* element has a single *Date* child element to specify the date and time of the record. The *Date* element uses the attributes in Table 1 to specify the date and time of the record. An evaluation was made by AODC for internal suitability as to whether or not to use the ISO8601 standard for specifying the date and time. It was decided that attributes would be used to specify the individual components of the date and time to allow the easy extraction of data from resultant XML files using XSLT. When defining an *Instant* using the *Date* child element, all of the attributes are required to be specified.

Name	Type	Description
year	Non-negative integer	Year value.
month	Non-negative integer	Month value between 1 and 12.
day	Non-negative integer	Day value between 1 and 31.
hour	Non-negative integer	Hour value between 0 and 23.
minute	Non-negative integer	Minute value between 0 and 59.
second	Non-negative integer	Second value between 0 and 59.
timeZone	String	Time zone value should be concurrent with a valid.

Table 1 Date Element Attributes

The *Period* element (Figure 7) can be utilised in a variety of ways. You may specify a continuous period, with specific instants in time for both the start and end of the period. You may also specify in-continuous periods that have start and end dates, and various sub-periods in-between. The *Period* element must contain *Start* and *End* child elements and may contain any number of inner *Period* child elements.

The *Start* and *End* elements may contain either an *Instant* or a *Date* child element. If an *Instant* is specified all of attributes of its child element *Date* must have values assigned. If a *Date* is specified without a parent *Instant* element, then the *Date* attributes are optional. This has been done to allow flexibility when defining in-continuous periods.

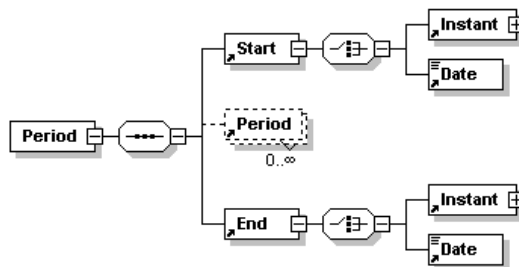


Figure 7 Period Element Structure

The *Period* element has an attribute “type” (Table 2). This is used to specify whether the period is continuous or periodical (contains inner periods).

Name	Type	Use	Description
type	Continuous or Periodical	Optional	This attribute should only be used in the outermost period element.

Table 2 Period Element Attributes

The following example defines a continuous period from 29/01/2002 11:50:00 to 29/01/2002 17:30:00.

```
<TemporalReference>
  <Period type="Continuous">
    <Start>
      <Instant>
        <Date year="2002" month="1" day="29" hour="11" minute="50" second="0" timeZone="Z" />
      </Instant>
    </Start>
    <End>
      <Instant>
        <Date year="2002" month="1" day="29" hour="17" minute="30" second="0" timeZone="Z" />
      </Instant>
    </End>
  </Period>
</TemporalReference>
```

The next example defines the period, all March, April and May data between 1995 and 2001.

```
<TemporalReference>
  <Period type="Periodical">
    <Start>
      <Instant>
        <Date year="1995"/>
      </Instant>
    </Start>
    <Period>
      <Start>
        <Date month="3"/>
      </Start>
      <End>
        <Date month="5"/>
      </End>
    </Period>
    <End>
      <Instant>
        <Date year="2001"/>
      </Instant>
    </End>
  </Period>
</TemporalReference>
```

3.3 Encoding Marine Data

The XML structure was developed with the flexibility to handle both point data and profile data. There are various ways to encode the same marine data within the XML structure. It is a matter of choosing the most beneficial way to suit the data type being encoded.

The *Data* element (Figure 8) encases a series of *DataObject* child elements that contain the data, along with information on the parameter measured and the sensor used.

Each *DataObject* element can be used to optionally specify the quality of the data it contains and to encapsulate data for one parameter or a series of parameters. This allows flexibility in the way in which data is represented within the XML structure. There is the option to place all the data for the record within one *DataObject* or in separate *DataObject* elements. It would be suggested that each individual *DataObject* is used to encode related data (for example data that is measured with the same sensor).

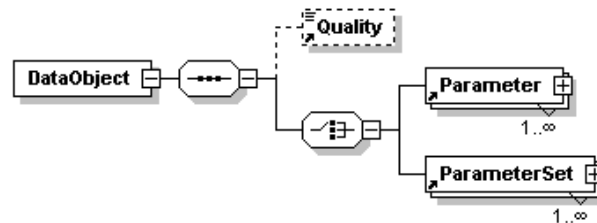


Figure 8 Data Element Structure

The *DataObject* element must have one or more *Parameter* child elements or one or more *ParameterSet* child elements. When using a series of *Parameter* child elements the values of the parameters are specified within individual *Value* tags. When using the *ParameterSet* to encode the data, the values of the parameters can be specified as a comma separated list of data or within individual *Value* tags. It is most idyllic to have the data values listed individually, however this can be impractical where a large number of parameters are observed for a *DataObject*. In this case, listing the individual values causes excessive markup which inturn renders actual data difficult to visually locate and the file size to balloon.

Each *Parameter* element (Figure 9) has two optional child elements, *Sensor* and *Value*. This element has compulsory attributes '*index*', '*name*' and '*units*'. The '*name*' and '*units*' attribute values can have the value of any string, however they should come from a list of valids for compatibility purposes. This issue is discussed in further detail later in this section.

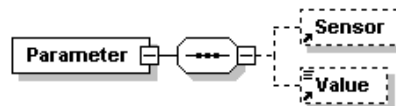


Figure 9 Parameter Element Structure

When specifying the *DataObject* element as a series of *Parameter* elements, the *Value* child element is compulsory. Another limitation of the DTD but not the SCHEMA is that this cannot be verified.

The *ParameterSet* element (Figure 10) is used to encapsulate a series of parameters, where that data will be encoded as a comma separated list. The most common use of this element would be to encapsulate profile data. It may contain an optional *Sensor* child element. This will be followed by one or

more *Parameter* child elements. Last of all it contains an optional *ValueList* child element.

There is a choice in the way in which you would like to encode data within a *ParameterSet* element. The child *Parameter* elements may be used to encode values within individual *Value* elements, or the child element *ValueList* may be used to specify a comma separated list of data. A limitation in the DTD and SCHEMA is that they cannot check whether the required *Value* elements or the *ValueList* element is present as they are both optional.

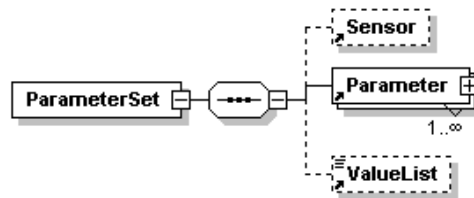


Figure 10 ParameterSet Element Structure

The *ValueList* element has an attribute '*numberOfValueSets*' which specifies the number of value sets contained within the value list. The list of values contained inside the *ValueList* tag should be made up of a series of sets of values for each parameter (in the order of the parameter indexes), separated by a comma. Each set of values should be separated by a space.

An example follows of how to specify simple point data using the *DataObject* and *Parameter* elements.

```
<DataObject index="0" type="Ancillary" numberOfParameters="1" reject="false">
  <Parameter index="0" name="Bathymetry" units="Metres">
    <Value>100.0</Value>
  </Parameter>
</DataObject>
```

Another example follows of how to specify simple point data and an associated flag value using the *DataObject* and *Parameter* elements.

```
<DataObject index="0" type="Primary" numberOfParameters="2" reject="false">
  <Parameter index="0" name="Sea Surface Temperature" units="Degrees Celcius">
    <Sensor name="XBT" model="Mk12 T-10" />
    <Value>25.3</Value>
  </Parameter>
  <Parameter index="1" name="Quality Control Flag" units="">
    <Value>1</Value>
  </Parameter>
</DataObject>
```

Another example of how to specify a temperature-depth profile and corresponding flag values using the *DataObject* and *ParameterSet* and *ValueList* elements.

Depth	Temperature	Flag
0	23.5	0
10	22.4	0
20	19.5	3

```
<DataObject index="0" name="XBT" type="Primary" numberOfParameterSets="1" reject="false">
  <ParameterSet index="0" numberOfParameters="3">
    <Sensor name="XBT" model="Mk12 T-10" />
    <Parameter index="0" name="Water Temperature" units="Degrees Celcius"/>
    <Parameter index="1" name="Water Depth" units="Metres" />
    <Parameter index="2" name="XBT-Mk12-AODC1.0" units="" />
    <ValueList numberOfValueSets="3">23.5,0,0 22.4,10,0 19.5,20,3</ValueList>
  </ParameterSet>
</DataObject>
```

To encode the same profile data above using the *DataObject*, *ParameterSet* and *Value* elements would make the file size significantly larger. This can be seen from the length of the example code below.

```
<DataObject index="0" name="XBT" type="Primary" numberOfParameterSets="3" reject="false">
  <ParameterSet index="0" numberOfParameters="3">
    <Sensor name="XBT" model="Mk12 T-10" />
    <Parameter index="0" name="Water Temperature" units="Degrees Celcius">
      <Value>23.5</Value>
    </Parameter>
    <Parameter index="1" name="Water Depth" units="Metres">
      <Value>0</Value>
    </Parameter>
    <Parameter index="2" name="XBT-Mk12-AODC1.0" units="">
      <Value>0</Value>
    </Parameter>
  </ParameterSet>
  <ParameterSet index="1" numberOfParameters="3">
    <Sensor name="XBT" model="Mk12 T-10" />
    <Parameter index="0" name="Water Temperature" units="Degrees Celcius">
      <Value>22.4</Value>
    </Parameter>
    <Parameter index="1" name="Water Depth" units="Metres">
      <Value>10</Value>
    </Parameter>
    <Parameter index="2" name="XBT-Mk12-AODC1.0" units="">
      <Value>0</Value>
    </Parameter>
  </ParameterSet>
  <ParameterSet index="2" numberOfParameters="3">
    <Sensor name="XBT" model="Mk12 T-10" />
    <Parameter index="0" name="Water Temperature" units="Degrees Celcius">
      <Value>19.5</Value>
    </Parameter>
    <Parameter index="1" name="Water Depth" units="Metres">
      <Value>20</Value>
    </Parameter>
    <Parameter index="2" name="XBT-Mk12-AODC1.0" units="">
      <Value>3</Value>
    </Parameter>
  </ParameterSet>
</DataObject>
```

Using *ParameterSet* and *ValueList* elements to encode profile data, although is not true XML, does create smaller file sizes that are easier to read and quicker to parse which outweighs the benefits gained from having the values of the data in individual tags. An example XML file is located in Appendix C that is based on Royal Australian Navy (RAN) collected data. This best illustrates how data is encapsulated in Marine XML.

3.4 Control Vocabularies

There are several instances within the XML structure where there are suggested valids for an element or attribute value. The valids were primarily drawn from NASA's Global Change Master Directory (GCMD) Directory Interchange Format (DIF) fields, which is the framework that metadata records created within the AODC adhere to. For more information on these valids, please visit <http://gcmd.gsfc.nasa.gov/>.

It is not intended to go into details of the definitions of the valids that the AODC use within this paper, suffice to say that they were developed to ensure the compatibility of our data sets internally. It should also be noted that a Marine XML file will still be validated by the Marine XML DTD or SCHEMA presented here even if it does not use the accepted valids. This allows flexibility in the use of Marine XML. To make data compatible between organisations the accepted valids would need to be used. If the data does not need to be compatible with external organisations then any value can be assigned.

3.5 Auditing

Creating an XML structure that can preserve edits is of significant benefit to data management within the AODC. The structure allows all of the data, original, flagged and edited, to be stored within the one file, along with other edit details, such as who performed the edit, where, when and why it was performed.

The *MarineDataSet* element has an optional child, *QCDetails*. This element can be used to specify any quality control tests and processes performed on the data, along with flag definitions. It also contains an optional *Edits* child element that can be used to track all edits performed on the data.

The *Edits* element (Figure 11) is used to encapsulate a series of individual edits performed on the data as *EditedValue* child elements.

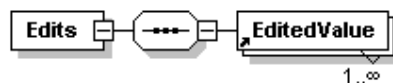


Figure 11 Edits Element Structure

Each *EditedValue* child element can capture information on the index of the record edited, the date of the edit, the editor, the Marine XML object edited, the name of the parameter/object edited, the previous value and the reason for the edit.

An example of the use of the *Edits* element to track edits within a data set can be seen in the example XML file in Appendix C.

3.6 Additional Features

The *Quality* element is used to specify the quality of data in a variety of locations throughout the XML structure. It is a compulsory child element of the root element, *MarineDataSet*. It is also an optional child element of the *MarineDataRecord*, *SpatialReference*, *TemporalReference*, and *DataObject* elements. In each case it can be used to specify the quality of the data encased by the element. Valid values for this element have been determined. The AODC presently only makes use of the compulsory *Quality* element, setting it to “Unknown” when data is originally encoded into XML and then changed to “Good” after completion of quality control procedures.

A very flexible element that is found in various locations throughout the XML structure is the *Property* element. This element can be used to set the name and value of any property. One example of the use of this element could be to encode metadata. The *MarineDataSet* element can have any number of *Property* child elements that could be used to encode any metadata. For example:

```
<Property name="Email">joe@joe.com.au</Property>
```

Another optional child element of *MarineDataSet* is the *Custodian* element. This has any number of *Property* child elements that can be used to specify any custodian details.

An optional child element of *MarineDataRecord* is the *Source* element. This can be used to assign details such as source file name, agency and project identification numbers, along with any details of the platform used.

4 Current Status

The Marine XML outlined in this paper is currently being used within the AODC to manage various types of RAN collected marine environmental data. Data types include temperature-depth profiles, sound velocity-depth profiles, meteorological data, seabed composition data, water transparency data and bioluminescence data.

MarineQC software has been developed as an interface to quality control the data stored in Marine XML. Data can be visually inspected and all edits performed using the software are consequently reflected in the Marine XML file. For more information about MarineQC please visit the AODC website at <http://www.aodc.gov.au>.

Metadata management has also been streamlined with XML. The Marine Environmental Data Information Referral Catalogue (MEDI), the AODC metadata management tool, has been modified to automatically generate DIF metadata records from marine data encoded in Marine XML.

5 Conclusion

The Marine XML presented in this paper is being used successfully within the AODC to encode marine data. The structure has the facilities to encapsulate profile and point data, with varying temporal and spatial extents. Details such as custodian, source and the quality of the data can be encoded. Edits can be tracked within the structure, features of the data can be flagged and format descriptions can be thrown away, making it a very useful tool in managing marine data.

The development of MarineQC, AODC quality control software, has been based around data encoded in Marine XML. This has simplified and accelerated the quality control procedures within the AODC, vastly improving our data processing capabilities. It has simplified the storage of marine data, by having all of the data, raw and edited, along with metadata (if necessary) within the one XML file for the whole of its life.

One disadvantage of using Marine XML to encode marine data is the large increase in file size. This can be overcome through the use of compression software such as WINZIP or GZIP. There are also several limitations in the capability of both the Marine XML DTD and SCHEMA to validate the structure of a Marine XML file. The DTD and SCHEMA have not been designed+ to check Marine XML elements and attributes against suggested valids. It was decided to keep a large amount of flexibility within the Marine XML structure by allowing the user to use a valid or to use any value. It must be noted however, where the choice is to not use valids, the dataset will not be compatible with other Marine XML data sets.

Future prospects of the use of Marine XML within the AODC are limitless. Stylesheets will be developed to display data graphically and be made available over the Internet. This will save having to manipulate the data itself into a variety of graphical formats. All manner of text, HTML and graphical reports based on statistics extracted from the data or the data itself can be developed using stylesheets that would grab the relevant information from the Marine XML files.

6 Comments

The AODC appreciates and welcomes any comments and reviews on this paper. Our Marine XML SCHEMA and/or DTD can also be made available upon request. Please direct all correspondence to dm@aodc.gov.au.

References

Searle, B., Technical Developments in Marine Data Management, Australian Oceanographic Data Centre, Australia.

Cox, S., et al, 2001, Geography Markup Language (GML) 2.0, OGC Recommendation Paper, OGC Document Number:01-029.

APPENDIX A Marine XML Document Type Definition (DTD)

```
<!ELEMENT MarineDataSet (Quality, Custodian?, QCDetails?, Property*, MarineDataRecord+)>
<!ATTLIST MarineDataSet
    caveat CDATA #IMPLIED
    name CDATA #IMPLIED
    description CDATA #IMPLIED
    creationDate CDATA #IMPLIED
>
<!ELEMENT Quality (#PCDATA)>
<!ELEMENT Custodian (Property+)>
<!ELEMENT QCDetails (Tests?, Processes?, Flags?, Edits?)>
<!ATTLIST QCDetails
    agency CDATA #IMPLIED
    operator CDATA #IMPLIED
    date CDATA #IMPLIED
    status (Complete | Partial | Unprocessed) #REQUIRED
>
<!ELEMENT Tests (Test+)>
<!ELEMENT Test (Result*)>
<!ATTLIST Test
    name CDATA #REQUIRED
    field CDATA #REQUIRED
    details CDATA #IMPLIED
>
<!ELEMENT Result EMPTY>
<!ATTLIST Result
    recordID CDATA #REQUIRED
    details CDATA #REQUIRED
>
<!ELEMENT Processes (Process+)>
<!ELEMENT Process EMPTY>
<!ATTLIST Process
    name CDATA #REQUIRED
    details CDATA #REQUIRED
>
<!ELEMENT Flags (FlagSet+)>
<!ELEMENT FlagSet (#PCDATA)>
<!ATTLIST FlagSet
    name CDATA #REQUIRED
>
<!ELEMENT MarineDataRecord (Quality?, Property*, SpatialReference, TemporalReference, Source, Data)>
<!ATTLIST MarineDataRecord
    ID CDATA #REQUIRED
    reject (true | false) #REQUIRED
>

<!ELEMENT Property (#PCDATA)>
<!ATTLIST Property
    name CDATA #REQUIRED
>
<!ELEMENT SpatialReference (Quality?, (GeoPoint | GeoBox | GeoLine | GeoPolygon))>
<!ELEMENT GeoPoint (Coordinates)>
<!ELEMENT GeoBox (Coordinates, Coordinates)>
<!ELEMENT GeoLine (Coordinates, Coordinates, Coordinates*)>
<!ELEMENT GeoPolygon (OuterBoundaryIs, InnerBoundaryIs*)>
<!ELEMENT OuterBoundaryIs (LinearRing)>
```

```

<!ELEMENT InnerBoundaryIs (LinearRing)>
<!ELEMENT LinearRing (Coordinates, Coordinates, Coordinates, Coordinates, Coordinates*)>
<!ELEMENT Coordinates (Latitude, Longitude, (Depth | Altitude)?)>
<!ATTLIST Coordinates
    datum CDATA #REQUIRED
>
<!ELEMENT Latitude (#PCDATA)>
<!ELEMENT Longitude (#PCDATA)>
<!ELEMENT Depth (#PCDATA)>
<!ATTLIST Depth
    datum CDATA #REQUIRED
    units CDATA #REQUIRED
>
<!ELEMENT Altitude (#PCDATA)>
<!ATTLIST Altitude
    datum CDATA #REQUIRED
    units CDATA #REQUIRED
>
<!ELEMENT TemporalReference (Quality?, (Instant | Period))>
<!ELEMENT Instant (Date)>
<!ELEMENT Date (#PCDATA)>
<!ATTLIST Date
    year CDATA #IMPLIED
    month CDATA #IMPLIED
    day CDATA #IMPLIED
    hour CDATA #IMPLIED
    minute CDATA #IMPLIED
    second CDATA #IMPLIED
    timeZone CDATA #IMPLIED
>
<!ELEMENT Period (Start, Period*, End)>
<!ATTLIST Period
    type (Continuous | Periodical) #IMPLIED
>
<!ELEMENT Start (Instant | Date)>
<!ELEMENT End (Instant | Date)>
<!ELEMENT Source (Platform?)>
<!ATTLIST Source
    isObservedData (true | false) #REQUIRED
    sourceFileName CDATA #REQUIRED
    agency CDATA #REQUIRED
    projectID CDATA #REQUIRED
>
<!ELEMENT Platform EMPTY>
<!ATTLIST Platform
    type CDATA #REQUIRED
    name CDATA #REQUIRED
    identifier CDATA #REQUIRED
>
<!ELEMENT Edits (EditedValue+)>

```

```

<!ELEMENT EditedValue EMPTY>
<!ATTLIST EditedValue
    recordID CDATA #REQUIRED
    date CDATA #REQUIRED
    editedBy CDATA #REQUIRED
    object CDATA #REQUIRED
    field CDATA #REQUIRED
    dataObjectIndex CDATA #IMPLIED
    parameterIndex CDATA #IMPLIED
    parameterSet CDATA #IMPLIED
    previousValue CDATA #REQUIRED
    reason CDATA #REQUIRED
>
<!ELEMENT Data (DataObject+)>
<!ATTLIST Data
    numberOfDataObjects CDATA #REQUIRED
>
<!ELEMENT DataObject (Quality?, (Parameter+ | ParameterSet+))>
<!ATTLIST DataObject
    index CDATA #REQUIRED
    name CDATA #IMPLIED
    type (Primary | Ancillary) #REQUIRED
    numberOfParameters CDATA #IMPLIED
    numberOfParameterSets CDATA #IMPLIED
    reject (true | false) #REQUIRED
>
<!ELEMENT Sensor EMPTY>
<!ATTLIST Sensor
    name CDATA #REQUIRED
    model CDATA #IMPLIED
    description CDATA #IMPLIED
>
<!ELEMENT Parameter (Sensor?, Value?)>
<!ATTLIST Parameter
    index CDATA #REQUIRED
    name CDATA #REQUIRED
    units CDATA #REQUIRED
>
<!ELEMENT ParameterSet (Sensor?, Parameter+, ValueList?)>
<!ATTLIST ParameterSet
    index CDATA #REQUIRED
    numberOfParameters CDATA #REQUIRED
>
<!ELEMENT Value (#PCDATA)>
<!ELEMENT ValueList (#PCDATA)>
<!ATTLIST ValueList
    numberOfValueSets CDATA #REQUIRED
>

```

APPENDIX B Marine XML Schema

```
<xs:schema >

  <xs:element name="Altitude">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:decimal">
          <xs:attribute name="datum" type="xs:string" use="required" />
          <xs:attribute name="units" type="xs:string" use="required" />
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>

  <xs:element name="Coordinates">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Latitude" />
        <xs:element ref="Longitude" />
        <xs:choice minOccurs="0">
          <xs:element ref="Depth" />
          <xs:element ref="Altitude" />
        </xs:choice>
      </xs:sequence>
      <xs:attribute name="datum" type="xs:string" use="required" />
    </xs:complexType>
  </xs:element>

  <xs:element name="Custodian">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Property" maxOccurs="unbounded" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="Data">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="DataObject" maxOccurs="unbounded" />
      </xs:sequence>
      <xs:attribute name="numberOfDataObjects" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>


```

```

<xs:element name="DataObject">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Quality" minOccurs="0" />
      <xs:choice>
        <xs:element name="Parameter" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element ref="Sensor" minOccurs="0" />
              <xs:element ref="Value"/>
            </xs:sequence>
            <xs:attribute name="index" type="xs:nonNegativeInteger" use="required" />
            <xs:attribute name="name" type="xs:string" use="required" />
            <xs:attribute name="units" type="xs:string" use="required" />
          </xs:complexType>
        </xs:element>
        <xs:element ref="ParameterSet" maxOccurs="unbounded" />
      </xs:choice>
    </xs:sequence>
    <xs:attribute name="index" type="xs:nonNegativeInteger" use="required"/>
    <xs:attribute name="name" type="xs:string" use="optional" />
    <xs:attribute name="type" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:NMTOKEN">
          <xs:enumeration value="Primary" />
          <xs:enumeration value="Ancillary" />
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="numberOfParameters" type="xs:nonNegativeInteger" use="optional" />
    <xs:attribute name="numberOfParameterSets" type="xs:nonNegativeInteger" use="optional" />
    <xs:attribute name="reject" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:NMTOKEN">
          <xs:enumeration value="true" />
          <xs:enumeration value="false" />
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>

<xs:element name="Date">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="year" type="xs:nonNegativeInteger" />
        <xs:attribute name="month" type="xs:nonNegativeInteger" />
        <xs:attribute name="day" type="xs:nonNegativeInteger" />
        <xs:attribute name="hour" type="xs:nonNegativeInteger" />
        <xs:attribute name="minute" type="xs:nonNegativeInteger" />
        <xs:attribute name="second" type="xs:nonNegativeInteger" />
        <xs:attribute name="timeZone" type="xs:string" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>

<xs:element name="Depth">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:decimal">
        <xs:attribute name="datum" type="xs:string" use="required" />
        <xs:attribute name="units" type="xs:string" use="required" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>

```

```

<xs:element name="EditedValue">
  <xs:complexType>
    <xs:attribute name="recordID" type="xs:nonNegativeInteger" use="required"/>
    <xs:attribute name="date" type="xs:string" use="required" />
    <xs:attribute name="editedBy" type="xs:string" use="required" />
    <xs:attribute name="object" type="xs:string" use="required" />
    <xs:attribute name="field" type="xs:string" use="required" />
    <xs:attribute name="dataObjectIndex" type="xs:nonNegativeInteger" />
    <xs:attribute name="parameterIndex" type="xs:nonNegativeInteger" />
    <xs:attribute name="parameterSet" type="xs:nonNegativeInteger" />
    <xs:attribute name="previousValue" type="xs:string" use="required" />
    <xs:attribute name="reason" type="xs:string" use="required" />
  </xs:complexType>
</xs:element>
<xs:element name="Edits">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="EditedValue" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="End">
  <xs:complexType>
    <xs:choice>
      <xs:element ref="Instant" />
      <xs:element ref="Date" />
    </xs:choice>
  </xs:complexType>
</xs:element>

<xs:element name="Flags">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="FlagSet" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="FlagSet">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="name" type="xs:string" use="required" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>

<xs:element name="GeoBox">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Coordinates" minOccurs="2" maxOccurs="2" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="GeoLine">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Coordinates" minOccurs="2" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="GeoPoint">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Coordinates" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

```

<xs:element name="GeoPolygon">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="OuterBoundaryIs" />
      <xs:element ref="InnerBoundaryIs" minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="InnerBoundaryIs">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="LinearRing" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="Instant">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Date">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="xs:string">
              <xs:attribute name="year" type="xs:nonNegativeInteger" use="required" />
              <xs:attribute name="month" type="xs:nonNegativeInteger" use="required" />
              <xs:attribute name="day" type="xs:nonNegativeInteger" use="required" />
              <xs:attribute name="hour" type="xs:nonNegativeInteger" use="required" />
              <xs:attribute name="minute" type="xs:nonNegativeInteger" use="required" />
              <xs:attribute name="second" type="xs:nonNegativeInteger" use="required" />
              <xs:attribute name="timeZone" type="xs:string" use="required" />
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="Latitude" type="xs:decimal" />

<xs:element name="LinearRing">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Coordinates" minOccurs="4" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="Longitude" type="xs:decimal" />

<xs:element name="MarineDataRecord">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Quality" minOccurs="0" />
      <xs:element ref="Property" minOccurs="0" maxOccurs="unbounded" />
      <xs:element ref="SpatialReference" />
      <xs:element ref="TemporalReference" />
      <xs:element ref="Source" />
      <xs:element ref="Data" />
    </xs:sequence>
    <xs:attribute name="ID" type="xs:nonNegativeInteger" use="required" />
    <xs:attribute name="reject" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:NMTOKEN">
          <xs:enumeration value="true" />
          <xs:enumeration value="false" />
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>

```

```

<xs:element name="MarineDataSet">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Quality" />
      <xs:element ref="Custodian" minOccurs="0" />
      <xs:element ref="QCDetails" minOccurs="0" />
      <xs:element ref="Property" minOccurs="0" maxOccurs="unbounded" />
      <xs:element ref="MarineDataRecord" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="caveat" type="xs:string" />
    <xs:attribute name="name" type="xs:string" />
    <xs:attribute name="description" type="xs:string" />
    <xs:attribute name="creationDate" type="xs:string" />
  </xs:complexType>
</xs:element>

<xs:element name="OuterBoundaryIs">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="LinearRing" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="Parameter">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Sensor" minOccurs="0" />
      <xs:element ref="Value" minOccurs="0" />
    </xs:sequence>
    <xs:attribute name="index" type="xs:nonNegativeInteger" use="required" />
    <xs:attribute name="name" type="xs:string" use="required" />
    <xs:attribute name="units" type="xs:string" use="required" />
  </xs:complexType>
</xs:element>

<xs:element name="Period">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Start" />
      <xs:element ref="Period" minOccurs="0" maxOccurs="unbounded" />
      <xs:element ref="End" />
    </xs:sequence>
    <xs:attribute name="type" use="optional">
      <xs:simpleType>
        <xs:restriction base="xs:NMTOKEN">
          <xs:enumeration value="Continuous" />
          <xs:enumeration value="Periodical" />
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>

<xs:element name="Platform">
  <xs:complexType>
    <xs:attribute name="type" type="xs:string" use="required" />
    <xs:attribute name="name" type="xs:string" use="required" />
    <xs:attribute name="identifier" type="xs:string" use="required" />
  </xs:complexType>
</xs:element>

<xs:element name="Process">
  <xs:complexType>
    <xs:attribute name="name" type="xs:string" use="required" />
    <xs:attribute name="details" type="xs:string" use="required" />
  </xs:complexType>
</xs:element>

```

```

<xs:element name="Processes">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Process" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="Property">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="name" type="xs:string" use="required" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>

<xs:element name="QCDetails">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Tests" minOccurs="0" />
      <xs:element ref="Processes" minOccurs="0" />
      <xs:element ref="Flags" minOccurs="0" />
      <xs:element ref="Edits" minOccurs="0" />
    </xs:sequence>
    <xs:attribute name="agency" type="xs:string" />
    <xs:attribute name="operator" type="xs:string" />
    <xs:attribute name="date" type="xs:string" />
    <xs:attribute name="status" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:NMTOKEN">
          <xs:enumeration value="Complete" />
          <xs:enumeration value="Partial" />
          <xs:enumeration value="Unprocessed" />
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>

<xs:element name="Quality" type="xs:string" />

<xs:element name="Result">
  <xs:complexType>
    <xs:attribute name="recordID" type="xs:nonNegativeInteger" use="required" />
    <xs:attribute name="details" type="xs:string" use="required" />
  </xs:complexType>
</xs:element>

<xs:element name="Sensor">
  <xs:complexType>
    <xs:attribute name="name" type="xs:string" use="required" />
    <xs:attribute name="model" type="xs:string" use="optional" />
    <xs:attribute name="description" type="xs:string" use="optional" />
  </xs:complexType>
</xs:element>

```

```

<xs:element name="Source">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Platform" minOccurs="0" />
    </xs:sequence>
    <xs:attribute name="isObservedData" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:NMTOKEN">
          <xs:enumeration value="true" />
          <xs:enumeration value="false" />
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="sourceFileName" type="xs:string" use="required" />
    <xs:attribute name="agency" type="xs:string" use="required" />
    <xs:attribute name="projectID" type="xs:string" use="required" />
  </xs:complexType>
</xs:element>

<xs:element name="SpatialReference">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Quality" minOccurs="0" />
      <xs:choice>
        <xs:element ref="GeoPoint" />
        <xs:element ref="GeoBox" />
        <xs:element ref="GeoLine" />
        <xs:element ref="GeoPolygon" />
      </xs:choice>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="Start">
  <xs:complexType>
    <xs:choice>
      <xs:element ref="Instant" />
      <xs:element ref="Date" />
    </xs:choice>
  </xs:complexType>
</xs:element>

<xs:element name="TemporalReference">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Quality" minOccurs="0" />
      <xs:choice>
        <xs:element ref="Instant" />
        <xs:element ref="Period" />
      </xs:choice>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="Test">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Result" minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="name" type="xs:string" use="required" />
    <xs:attribute name="field" type="xs:string" use="required" />
    <xs:attribute name="details" type="xs:string" use="optional" />
  </xs:complexType>
</xs:element>

<xs:element name="Tests">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Test" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

```

<xs:element name="Value" type="xs:string" />

<xs:element name="ParameterSet">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Sensor" minOccurs="0" />
      <xs:element ref="Parameter" maxOccurs="unbounded" />
      <xs:element ref="ValueList" minOccurs="0" />
    </xs:sequence>
    <xs:attribute name="index" type="xs:nonNegativeInteger" use="required" />
    <xs:attribute name="numberOfParameters" type="xs:nonNegativeInteger" use="required" />
  </xs:complexType>
</xs:element>

<xs:element name="ValueList">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="numberOfValueSets" type="xs:nonNegativeInteger" use="required" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>

</xs:schema>

```

APPENDIX C Example Marine XML File

```

<MarineDataSet caveat="Unclassified" name="Mk12XBT" description="Dataset contains temperature-depth profiles, along
with bathymetry." creationDate="2002-21-22T17:24:15K">
  <Quality>Good</Quality>
  <Custodian>
    <Property name="Agency">Australian Oceanographic Data Centre</Property>
    <Property name="WebSite">http://www.metoc.gov.au/</Property>
  </Custodian>
  <QCDetails agency="AODC" operator="LT" date="2001-05-22T12:47:17K" status="Complete">
    <Tests>
      <Test name="DuplicateCheck1.0" field="SpatialReference and TemporalReference" />
      <Test name="RepeatObsCheck1.0" field="SpatialReference and TemporalReference" />
      <Test name="RangeCheck1.0" field="SpatialReference and TemporalReference" />
      <Test name="SpeedCheck1.0" field="SpatialReference and TemporalReference">
        <Result recordID="0" details="Average speed to observation 1 is 112 knots. This exceeds the upper speed
limit of 30 knots." />
      </Test>
      <Test name="LandCheck1.0" field="SpatialReference" />
    </Tests>
    <Flags>
      <FlagSet name="XBT-Mk12-AODC1.0 Flags"> 0=No Flag(Accept); 1=Hit Bottom(Reject); 2=Wire Break;
3=Wire Stretch(Accept); 4=Wire Stretch(Reject); 5=Leakage(Accept); 6=Leakage(Reject);
7=Spikes(Accept); 8=Spikes(Reject); 9=Constant Temperature Profile(Accept); 10=Constant
Temperature Profile(Reject);11=Insulation Penetration(Accept); 12=Insulation
Penetration(Reject);13=Temperature Offset(Reject);</FlagSet>
    </Flags>
    <Edits>
      <EditedValue recordID="0" date="2001-05-22T12:47:15K" editedBy="AODC.LT"
object="SpatialReference.GeoPoint.Coordinates" field="Longitude" previousValue="115.281667"
reason="Failed SpeedCheck1.0" />
    </Edits>
  </QCDetails>
  <MarineDataRecord ID="0" reject="false">
    <SpatialReference>
      <GeoPoint>
        <Coordinates datum="WGS84">
          <Latitude>-34.553333</Latitude>
          <Longitude>151.281667</Longitude>
        </Coordinates>
      </GeoPoint>
    </SpatialReference>
    <TemporalReference>
      <Instant>
        <Date year="2001" month="3" day="28" hour="5" minute="40" second="13" timeZone="Z" />
      </Instant>
    </TemporalReference>
    <Source isObservedData="true" sourceFileName="T0_00001.RDF" agency="RAN" projectID="01004AN">
      <Platform type="MARINE" name="HMAS ANZAC" identifier="VKNG" />
    </Source>
    <Data numberOfDataObjects="3">
      <DataObject index="0" name="XBT" type="Primary" numberOfParameterSets="1" reject="false">
        <ParameterSet index="0" numberOfParameters="3">
          <Sensor name="XBT" model="Mk12 T-10" />
          <Parameter index="0" name="Water Temperature" units="Degrees Celcius"/>
          <Parameter index="1" name="Water Depth" units="Metres" />
          <Parameter index="2" name="XBT-Mk12-AODC1.0 Flags" units="" />
          <ValueList numberOfValueSets="2"> 22.874,0.63,0 22.936,1.26,0</ValueList>
        </ParameterSet>
      </DataObject>
      <DataObject index="1" type="Primary" numberOfParameters="1" reject="false">
        <Parameter index="0" name="Sea Surface Temperature" units="Degrees Celcius">
          <Value>21.0</Value>
        </Parameter>
      </DataObject>
      <DataObject index="2" type="Ancillary" numberOfParameters="1" reject="false">
        <Parameter index="0" name="Bathymetry" units="Metres">
          <Value>100.0</Value>
        </Parameter>
      </DataObject>
    </Data>
  </MarineDataRecord>
</MarineDataSet>

```