NetCDF Record Format And Real-Time Broadcast Protocol Specifications

by

Fabrice Lefebvre (SHOM), Gaël Quéméner (Ifremer)

Catherine Maillard, Armel Rué, Jean-Marc Sinquin, Michel Voisset (Ifremer)

Pascal Viollette (Genavir)

Copyright © 2003 SHOM - Ifremer

DNIS/ESI/SIR/DTI/02-246

Contents

Chapter 1 Introduction

In a general point of view, one's can say that the software architecture on board the vessels is made of acquisition servers, one recording client, and others secondary clients. This type of architecture is based on three technical specifications :

- The physical and logical connection between servers and clients.
- The formats of messages exchanged between servers and clients.
- The formats of logged data.

This document specifies these three technical points.

1.1 Connection between server(s) and client(s)

Servers and clients do communicate by standard IP services. The acquired data flow is transmitted by UDP in broadcast mode (cf. *[UDP]* et *[Broadcast]*), and some specific exchanges between servers and clients are based on TCP (cf. *[TCP]*), following a simple request/reply mode.

1.2 Format of network messages

The transmitted messages in network packets use an XML syntax and UTF-8 encoding.

1.3 Data recording format

Data are logged in NetCDF files. NetCDF files are binary files, convenient for a big amount of data. It is also convenient for further access to data. The multi-OS API to read and write NetCDF files is also an advantage.

1.4 Notations

Equipment An equipment is a set of hardware and software which transforms physical measures to numeric data. They communicate with an acquisition server by non standard hardware and software link (although most of them communicate via RS 232 and NMEA messages).

Acquisition time Time at which one information is received by an acquisition server, measured with the reference time system of the server. As transmission duration is significant, it is specified that the acquisition time is the time when the acquisition server received the beginning of the information. The time references of acquisition servers must be synchronized in real-time if data from both servers have to be merged.

Measure time Time associated with a measure. This time is given by the equipment or by removing a constant offset from the acquisition time.

Broadcast message A broadcast message is an XML message with numeric data and broadcast by an acquisition server through UDP

Description message A description message is an XML message sent to an identified client. It contains the detailed description of one broadcast message.

Time reference Each workstation on board a vessel has an internal time reference. It can be synchronized to an external master time reference by NTP protocol. The external master time reference is usually synchronized with UTC time by a GPS system.

Description request A description request is an XML message sent by a client to an acquisition server. It contains the message Id for which the client wants the description.

Acquisition server An acquisition server acquire data from one or many equipments to which it is connected. Data are broadcast on the network in XML broadcast messages.

Frame A frame is a component of a broadcast message. It contains a set of data with one common acquisition time.

1.5 Documents

BIPM	The International System of Units (SI). 7th edition, 1998. Organisation Intergouvernementale de la Convention du Mètre. See http://www.bipm.org/enus/6_Publications/si/si-brochure.html
CF	NetCDF Climate and Forecast (CF) Metadata Conventions. See http://www.cgd.ucar.edu/cms/eaton/netcdf/CF-current.htm
Namespaces	Recommandations W3Cfor XML namespaces. See http://www.w3.org/TR/1999/REC-xml-names-19990114
NetCDF	Documentation NetCDF. See http://www.unidata.ucar.edu/packages/netcdf/guidec
RFC 1035 (NTP)	Network Time Protocol (version 3). See http://www.faqs.org/rfcs/rfc1305.html
RFC 768 (UDP)	User Datagram Protocol. See http://www.faqs.org/rfcs/rfc768.html
RFC 791 (IP)	Internet Protocol. See http://www.faqs.org/rfcs/rfc791.html
RFC 793 (TCP)	Transmission Control Protocol. See http://www.faqs.org/rfcs/rfc793.html
RFC 894	A standard for the transmission of IP datagrams over Ethernet networks.. See http://www.faqs.org/rfcs/rfc894.html
RFC 919 (Broadcast)	Broadcasting internet datagrams. See http://www.faqs.org/rfcs/rfc919.html
RFC922	Broadcasting internet datagrams in the presence of subnets. See http://www.faqs.org/rfcs/rfc922.html
SI rules	Guide for the Use of the International System of Units (SI). NIST special publication 811. See http://physics.nist.gov/Document/sp811.pdf
UdUnits	Document UdUnits. See http://www.unidata.ucar.edu/packages/udunits
XML	Recommandations W3C for XML. See http://www.w3.org/TR/2000/REC-xml-20001006
XSD	Recommandations W3C for XSD. See http://www.w3.org/XML/Schema

Chapter 2 General concepts

2.1 Broadcasting on network

The data acquired by an acquisition server are broadcast at regular time-interval by an acquisition server on the ship local network by identified *broadcast messages.*

The broadcast frequency is usually the same as the acquisition frequency. An acquisition server does normally just encode data to the broadcast format without any heavy calculation or analysis.

The broadcast messages sent by an acquisition server are received by one or many clients. Broadcast messages are not self descriptive in order to limit the size of messages and the use of network capabilities. Any client can ask the server by a *description request* to give a description message giving details about the *broadcast messages* format.

An acquisition server periodically broadcast its configuration by UDP broadcast messages on a fixed and known IP port. The acquisition server configuration contains information about equipments, broadcast addresses, addresses used for description requests. Client programs which need to communicate with the server have to used these addresses.

2.2 Messages formats

Each **data broadcast message** contains one or many **frames**. Each of these frames are identified and time-stamped by the acquisition server. All the data contained in one frame have therefore the same time-stamp (there can be an exception, but it is rarely used, see section 5.5 Data Time-stamping). If the measure time is known, the acquisition server gives it.

The description messages contain :

- the necessary meta-data to decode data (units, data type, display format, valid minima and maxima, description, etc.) ;
- each frame description;
- the necessary meta-data to deal with messages configuration.

It would have been technically possible to transmit meta-data with each data broadcast messages. But that would have been to heavy and would have used too much of the network resources. So a two-phase mechanism has been chosen.

2.3 Recording

There is one client responsible for recording the data broadcast by all the acquisition servers of the network. This client “translate” in NetCDF files all the data contained in XML frames. NetCDF meta-data are extracted from description messages.

The record client creates as many NetCDF files as there are broadcast frames. Each received frame are then recorded in its proper file.

One variable in a NetCDF file corresponds to one data in one frame. Each NetCDF file contains a special « `time` » variable which contains the acquisition time. Each variable attributes correspond to source data description. The NetCDF file global attributes describe the acquisition server, the record process and the equipment.

Chapter 3 Network Protocol

3.1 Communication protocol

The acquisition server periodically broadcast (every 10 seconds, UDP protocol) **on the IP port number 4000** one or many configuration messages (cf. p. 8). Each configuration message contains equipments identifications, their messages, their source addresses, and their request addresses. A client which does not receive any configuration message after one minute can consider that the equipment does not exist any more.

This mechanism making distinction between broadcast sources and description source allows, for example, to run a description message server apart from the acquisition server (lighter code). This description message server will then be based on a data dictionary.

Any client program which wants to get a data broadcast message sent by an acquisition server will use one of the addresses defined by this server. Configuration messages and data messages are transmitted in UDP broadcast, so that the servers work without taking care of the clients. Description messages are sent through the TCP/IP protocol, as a reply to a request message.

A client can send a description request (cf. p. 14) by a TCP/IP connection to one of the address specified by the server. The destination program has to reply within 5 seconds by a description message as a reply to the connected client (cf. p. 15). If the client has not received the description within 5 seconds, it can send its request again or acts as it wants, as long as its behavior is specified.

It is highly recommended to limit the description request in order not to overload the servers. It is convenient to send one description request for each message when starting the client program. Sending a description request after each data message received has to be banned.

Client do know the *maximal period* of broadcast messages, by their own configuration or by reading the corresponding attribute in the description message. When a client has not have received any data after a much greater time than this maximal period, it can consider the server as non-existent and run a specific appropriate action.

We have to notice that an acquisition server that does not receive data from an equipment has to send an information about that. It has to emit a periodic message stating that there are no data so that the clients make the difference between a problem with the acquisition server and a problem on an equipment.

> In a practical point of view, many servers use the same address for broadcasting data. They must use the device id of the equipments to identify thei data messages. The same remark has to be applied to the clients.

3.2 Maximal size of UDP packets

[UDP] does not define any limit for the size of its packets; the only limit is the two bytes used in the header to store the size, which is 65535 characters minus the size of the IP and UDP headers.

Though, it is possible that the IP layer gives a limit to the broadcast packets size (MTU), even if IP protocol can fragment bigger packets. In a practical point of view, the maximum size of broadcast packets depends on operating systems and switches. Modern systems and network equipment seem to allow broadcast packets up to 65507 characters. We though recommend to limit their size to 1000 bytes and then to test the transmission in the case of higher size.

The problem of the packet size does just concern UDP/broadcast packets, other messages are transmitted through point to point TCP connections (cf. *[TCP]*). Broadcast messages are generally much less than 1000 bytes but it could be more than 1500 bytes (MTU).

Chapter 4 Configuration messages

4.1 Structure of a configuration message

```
default namespace ="http://www.ifremer.fr/flotte/TECHSAS/2002"
datatypes xsd = "http://www.w3.org/2001/XMLSchema-datatypes"

start = root

root = element configuration {
      element device {
```

This element describes a device. All given attributes are saved as global attributes in the FXND NetCDF file.

```
            attribute deviceid        { devid } ,
            attribute devicename      { nonemptystring } ,
```

Usual name of the device. No unicity constraint is associated with this name, which should simply allow a user to easily identify the concerned device.

```
            attribute firstusedate    { xsd:dateTime } ?,
```

Date of first use of the device, whatever this means for the organization responsible of its use.

```
            attribute position        { xsd:string } ?,
```

Indicate the room number of the device if it is onboard, and simply "towed" if it is towed.

```
            attribute x               { xsd:float } ,
```

Relative position of the main sensor of the device relative to the reference point of the ship in the X direction (axis along the heading, positive values are forward)

```
            attribute y               { xsd:float } ,
```

Relative position of the main sensor of the device relative to the reference point of the ship in the Y direction (horizontal axis transverse to the X axis, positive to the right)

```
            attribute z               { xsd:float } ,
```

Relative position of the main sensor of the device relative to the reference point of the ship in the Z direction (vertical axis, positive values are below the reference point)

```
            attribute latestcalibrationdate
      {xsd:dateTime} ?,
```

Date of the latest calibration of the device. This attribute is optional, but may be required for devices subject to drift and whose calibration parameters are dated.

```
            attribute calibrationparameters
      {nonemptystring}?,
```

This attribute is left open for devices drivers to specify calibration parameters. This attribute may be required for some devices if their data is to be post-processed.

```
            attribute workingparameters
      {nonemptystring}?,
```

This attribute is left open for devices drivers to specify working parameters (or parameters used when the device was configured). This attribute may be required for some devices if their data is to be post-processed. Note that even GPS post-processing may benefit from some meta-data such as the datum used by the GPS or any other meta-data which everybody takes for granted but which are better written down.

```
            attribute sourcetypelist { list { name+ } } ,
```

List of source types provided by this device. Source types should be separated with a space. For instance, a POS/MV may be a GPS position source and an attitude source, and this is represented by "gps att"

```
            attribute installdate     { xsd:dateTime } ?,
```

Date of installation (used only for permanent devices)

```
            element message                   { messageid, description ? } +,
            element broadcastaddress          { networkaddress } +,
            element descriptionaddress        { networkaddress } +
      } +
}

messageid      =  attribute name {xsd:string}, version
description    =  attribute description { nonemptystring }
networkaddress =  attribute host      {xsd:string} ,
                  attribute port      {xsd:nonNegativeInteger},
```

RFC2396 hostname or IP address

Network port number

```
                attribute protocol { "udp" | "tcp" } ?

version        =  attribute major    { xsd:nonNegativeInteger } ,
                  attribute minor    { xsd:nonNegativeInteger }

name  = xsd:string { pattern = "[A-Za-z][0-9A-Za-z_]*" minLength = "1" maxLength =
"30"}
devid = xsd:string { pattern = "[0-9A-Za-z_]+" minLength = "1" maxLength = "40" }
nonemptystring = xsd:string { minLength = "1" }
```

Formal specification in RELAX NG syntax of a configuration message

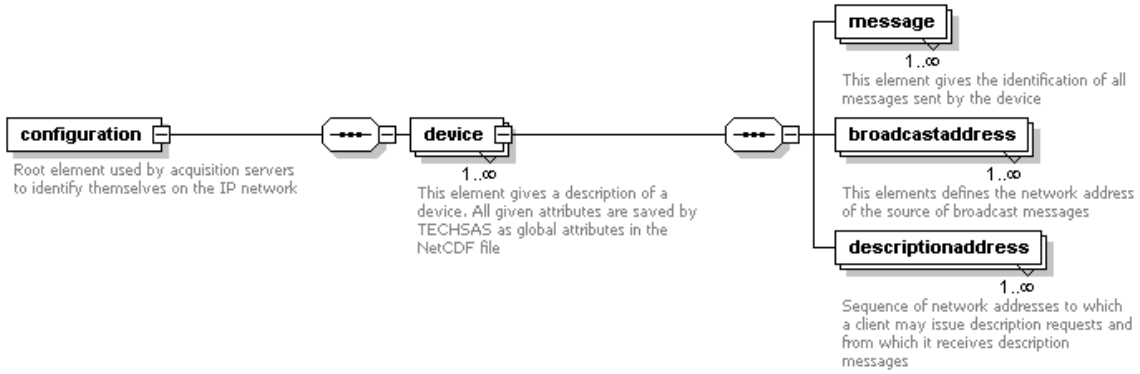

Figure 1 : graphic presentation of the structure of a configuration message

A configuration message is composed by :

- a root element (**<configuration>**) ;
- an equipment list (**<device>**) ;
- for each equipment :
- a list of broadcast messages (**<message>**) sent by the acquisition server for the equipment. All the listed messages are not necessarily sent at the same time ;
- a list broadcast addresses (**<broadcastaddress>**). Each of the messages can be sent to any of the broadcast addresses ;
- a list of addresses to be used for sending description request (**<descriptionaddress>**) (TCP protocol) ;

4.2 Example of a configuration message

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration xmlns="http://www.ifremer.fr/flotte/TECHSAS/2002">
    <device deviceid="Aquarius01"
            devicename="Aquarius GPS"
            sourcetypelist="gps"
            firstusedate="2001-09-10T10:30:00"
            position="bridge"
            x="15"
            y="0.23"
            z="-8.75"
            latestcalibrationdate="2001-09-10T10:30:00Z"
            calibrationparameters="no calibration required"
            installdate="2001-09-10T10:30:00Z"
            workingparameters="WGS84">
        <message name="GPSfix" major="1" minor="0"/>
        <broadcastaddress host="192.168.25.2" port="10000"/>
```

```
        <descriptionaddress host="192.168.25.2" port="11" protocole="udp"/>
    </device>
</configuration>
```

4.3 Rules for building a configuration message

Each acquisition server builds its configuration message with information from its own configuration.

The acquisition software operator in charge of the acquisition servers configuration that broadcast data on the network. It has to ensure that:

- there are not two equipments on board with the same **deviceid** attribute;
- the values of the **devicename** and **position** attributes follow the naming rules of the ship or company ;
- the ports and addresses used by the protocol are not reserved for any other service.

The administrator of the acquisition server has to ensure that :

- the value of a message **name** attribute is either a name of the *data dictionary* or a name which cause no conflict with a message name declared in the dictionary. In order to avoid such conflicts, it is highly recommended to prefix the dictionary unregistered messages by `Unregistered` ;
- If the value of the **name** attribute of a message is registered in the dictionary, then the **msgMajor** et **msgMinor** version numbers must be also registered in the dictionary ;
- The source types of the **sourcetypelist** attributes are all registered in the dictionary (no exception is possible) ;
- Calibration and working parameters are properly entered in the acquisition server configuration files or received from the equipment.

An acquisition server that emits many different data broadcast messages can emit many different configuration messages or merge all the information in one unique message.

4.4 Rules to process a configuration message

Any destination client program has to be able to deal with an XML message which does not conform to the specification, or with any non XML message.

Any client receiving a configuration message, has to keep these informations in memory, the **deviceid** attribute being the primary access key of these informations, and the messages **name**, **major**, **minor** attributes being the secondary keys. The network addresses are then used to initialize the communication protocol between the client program and the acquisition server.

It is possible to configure a client so that it only deals with one or many different **deviceid**, **name**, or **sourcetypelist** (etc.), or a particular combination of these simple filters.

Chapter 5 Broadcast messages

5.1 Structure of a broadcast message

The following message structure represents a typical broadcast message:

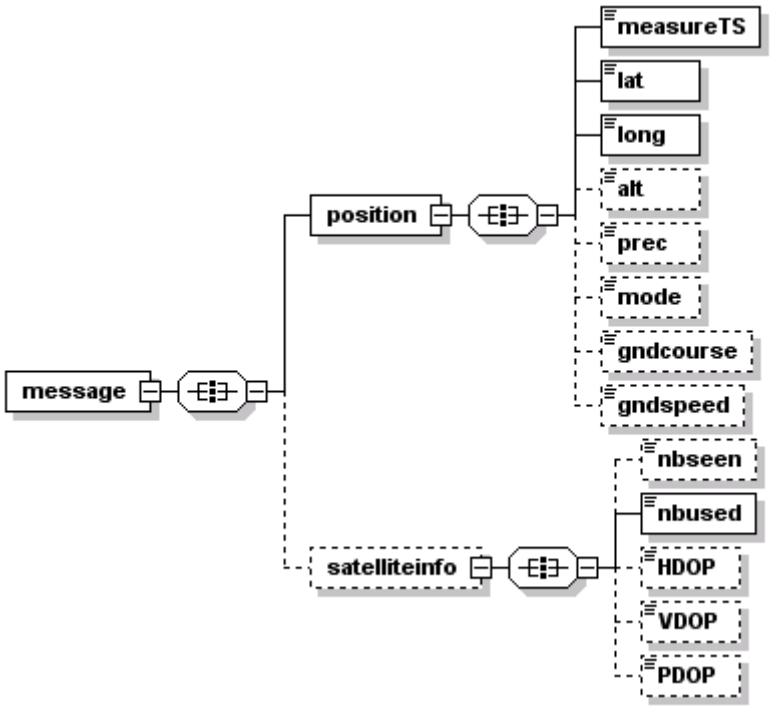

Figure 2 : graphic presentation of a broadcast message

A broadcast message is composed by :

- a **<message>** root element, with one or more elements describing the frames, each of these frame elements containing one or many elements describing data. The elements which are not identified by the description message are ignored ;
- the **<message>** root element must have the **name**, **major**, **minor**, **deviceid** attributes. Any other attribute is ignored ;
- each frame must have an **acquisitiontime** attribute giving the acquisition time of data in the frame. Any other attribute is ignored ;
- data can have other attributes which will be ignored ;
- data must not have child elements ;
- data values are encoded as nodes of type "text".

5.2 Example of a broadcast message

```
<?xml version="1.0" encoding="UTF-8"?>
<message name="GPSfix" major="1" minor="0" deviceid="Aquarius01">
    <position acquisitiontime="37279.6411858415" IgnoredAttribute="foo">
        <measureTS>37279.6411858415</measureTS>
        <long IgnoredAttribute="nothing of interest">44.924351</long>
        <lat>-4.934501</lat>
        <gndcourse>96.2</gndcourse>
        <gndspeed>5.7</gndspeed>
    </position>
    <satelliteinfo acquisitiontime="37279.6411858415">
        <nbused>8</nbused>
        <nbseen>10</nbseen>
        <HDOP>4.5</HDOP>
    </satelliteinfo>
</message>
```

5.3 Rules for building a broadcast message

In order to send the less amount of data as possible and to avoid IP packet fragmentation, a special attention is paid so that the size of broadcast messages is as small as possible. One's can follow the following considerations :

- The XML messages are encoded in UTF-8 and the XML header of the message indicates it: `<?xml version="1.0" encoding="UTF-8"?>`;
- Broadcast messages are sent with no specified name space.
- Space characters and carriage return can be erased when possible ;
- When a data element has got a default value, it is possible to send this element with no data. For instance, if the **<angle>** element has the 0 default value, then it is possible to send this data by a simple `<angle/>` rather than `<angle>0</angle>` ;

The **acquisitiontime** attribute of a broadcast frame has the ***xsd:dateTime*** type, i.e. a character string with *YYYY-MM-DD*T*HH*:*MM*:*SS.SSS*Z syntax. The other dates and times to be sent are encoded in such a way that they can be understand with the *[UdUnits]* library. The chosen unit for encoding dates is `days since 1899-12-30 00:00:00 UTC` because these dates are directly processed by Microsoft Excel software without any need of pre-conversion.

Numbers are written conforming their type specification.

In order to indicate a client that one or many data are missing in the case of a communication problem between the equipment and the acquisition system, the acquisition system has two possibilities: or it does not send the data element, or it sends an empty frame.

An acquisition program must check the syntax of the XML messages it broadcasts.

5.4 Rules for decoding a broadcast message

In order to ensure a correct decoding of messages whatever the future evolutions of these messages will be, it is important to follow these rules :

- Any destination client program has to be able to deal with an XML message which does not conform to the specification, or with any non XML message.
- The order of frames in a message has no importance. Neither is the order of data elements in a frame. These elements are identified by their name ;
- As an exception to the preceding rule, a data element with a **cardinalitymax** attribute greater than one, can be repeated more than once in a frame. In this case, the relative position of each element is significant ;

- A client must check the message **name**, **major**, **minor** attributes before accepting it. It should be able to accept without any difficulty a message which **minor** attribute is different from the one expected ;
- The character strings have to be processed according to the XML encoding, even if it is UTF-8 ;
- an empty data element is equal to its default value (**defaultvalue**), specified in the description message ;
- a missing data element is considered equal to its missing value (**missing_value**), specified in the description message ;
- A client is free to give its own meaning to an empty frame : for example, a client displaying periodic data can consider it as a "communication error" rather than displaying the missing values.

5.5 Data Time-stamping

In a general point of view, there is a difference between acquisition time and measure time.

Measure time is the time when a data has been sampled and digitalized in one equipment. This time has often got a bad precision because the sampling / integration / digitalization process in an equipment has a duration which can not be neglected. Moreover the measure time can be chosen at any time of the process, and that time is never specified in the technical documentations.

The acquisition time is only available on the acquisition server and is set to the time of reception of the numeric data from the equipment. If the equipment and the acquisition server are synchronized, the acquisition time is posterior to the measure time.

Acquisition time of data are always available and is transmitted in the **acquisitiontime** attribute of each frame. The generally adopted convention says that the acquisition time sent is the lowest of all the data acquisition frame. Measure times are not always available.

In some cases, an acquisition server will be able to acquire different single data from one equipment at different times and then broadcast them in one unique frame in a message. The acquisition time of each single data can then be transmitted in an single data element identified by the **acquisitiontimedata** attribute of the data element which needs a single acquisition time.

In other cases, an equipment can transmit one or many measure times for different data. Each of these times can be transmitted in a data element identified by the **measuretimedata** attribute.

In any case, the specifications of each acquisition server must specify precisely its time-stamping process.

Chapter 6 Description requests

6.1 Request structure

```
default namespace ="http://www.ifremer.fr/flotte/TECHSAS/2002"
datatypes xsd = "http://www.w3.org/2001/XMLSchema-datatypes"

start = root

root = element descriptionrequest { messageid }

messageid      =  attribute name {xsd:string}, version

version        =  attribute major    { xsd:nonNegativeInteger } ,
                  attribute minor    { xsd:nonNegativeInteger }
```

Formal specification in RELAX NG syntax of a request description structure

A description request contains one unique **<descriptionrequest>** root element that contains the useful elements for identifying the message.

6.2 Example of request

```
<?xml version="1.0" encoding="UTF-8"?>
<descritionrequest xmlns="http://www.ifremer.fr/flotte/TECHSAS/2002"
                   name="GPSfix" major="1" minor="0" />
```

6.3 Rules for building a description request

The **name**, **major**, **minor** attributes of the **<descriptionrequest>** element in a request description are the same as the ones in the broadcast configuration message.

The root element is emitted in the http://www.ifremer.fr/flotte/TECHSAS/2002 spacename.

6.4 Rules for replying to a description request

Any client program has to be able to deal with an XML message which does not conform to the specification, or with any non XML message.

When a description request is received (usually by an acquisition server), the **name**, **major**, **minor** attributes of the **<descriptionrequest>** element are compared to those saved in memory. In case of equality, the corresponding description message is sent. In case of non equality, nothing is done.

It is possible to create a program which would be able to deal with many different description requests and to reply by the correct description. For example, a program can look for any message description in the dictionary and reply to any description request.

Chapter 7 Description messages

7.1 Structure of a description message

```
default namespace ="http://www.ifremer.fr/flotte/TECHSAS/2002"
datatypes xsd = "http://www.w3.org/2001/XMLSchema-datatypes"

start = root

root = element messagedescription {
      messageid                                          ,
      description                                      ?,
      element frame {
      frameid                                                 Name and version of a frame as registered in the data
                                                              dictionary. A frame which has not been registered must still
                                                              have a unique name among all frame names for a given
                                                              sourcetype.
      attribute sourcetype          { nonemptystring }        Type of source for this frame. The list of possible source types
                                                              is registered in the data dictionary. It is still possible to use an
                                                              unregistered source type if it is prefixed by the letters "U_".
                                                              This means that no registered source type may begin with
                                                              these letters.
      description                                             Human readable description of the information in the message
      attribute period              {xsd:float}               Maximum time interval between two frames, in seconds
      attribute mandatory           { "true" | "false" } ,
      element data {

           dataid                                             Data element name and version registered in the data
                                                              dictionary

           description                                        Intended meaning of the data element regardless of its pattern
                                                              of usage
           attribute units          { nonemptystring }        The units attribute is required for all variables that represent
                                                              dimensional quantities. The value of the units attribute is a
                                                              string that can be recognized by UNIDATA's Udunits package.
                                                              Note that case is significant in the units strings. The Udunits
                                                              package defines a few dimensionless units, such as percent,
                                                              but is lacking commonly used units such as ppm (parts per
                                                              million). We recommend that the units attribute still be used
                                                              for these types of units, but have not attempted to standardize
                                                              any new dimensionless units. On the other hand there is no
                                                              need for a wide variety of dimensionless units for quantities
                                                              like sea-ice concentration, cloud fraction, probability and so
                                                              on; this descriptive information should be given in the
                                                              longname attribute rather than the units. As a general rule of
                                                              thumb, CF conventions should be adopted for this attribute.
           attribute shortunits     { nonemptystring }        Contains a short version of the units attribute, used for display
                                                              purposes. For instance, units may be constrained by UdUnits
                                                              and the dictionary to contain a "degree_Celsius" value,
                                                              whereas the shorter form "°C" may be preferable for displays.
                                                              When this attribute is not given, then the value found in units
                                                              shall be used. This attribute shall not be used for formal
                                                              conversion purposes.
           attribute valid_min      { xsd:double }            Minimum and maximum valid value for this data element. If
           attribute valid_max      { xsd:double }            these attributes are not assigned, then the minimum and
                                                              maximum valid values are the respective minimum and
                                                              maximum value representable with the underlying data type.
           attribute missing_value { xsd:string }             Value stored when no value is sent
           attribute type  {  "byte" | "char" |               Underlying physical data type recommended for the storage of
                              "short" | "int" |               values of the data element. The list of available data types is
                              "float" | "double" }            directly linked to the list of data types available in NetCDF.
                                                              The "char" type is reserved for character string storage. All
                                                              numeral data types are signed, except for the byte data type
                                                              which may be considered unsigned if validmin is 0 and above
                                                              and validmax is between 128 and 255.
           attribute positive       { "up" | "down" }         Required if the vertical axis units is not upward when
                                                              displaying the data in graphs (otherwise it is optional).
           attribute broadcastname { name }                   Name of the data element as it should appear in the real time
                                                              XML data flow or in FXND NetCDF files. No data element
                                                              may have the name "time" which is reserved.
           attribute long_name      { nonemptystring }        A long descriptive name. This could be used for labeling plots,
                                                              for example. If a data element has no long_name attribute
                                                              assigned, then element_name or broadcastname may be used..
```

attribute *cardinalitymin* {xsd:nonNegativeInteger}, attribute *cardinalitymax* {xsd:nonNegativeInteger},	*Minimum and maximum number of occurrences of this data element in a single frame. A data element whose cardinalitymin is 0 is optional. An optional data missing from a frame is considered having the value defined by the missing_value attribute.*
attribute *scale* { xsd:nonNegativeInteger }	*Maximum number of significant digits (decimal notation) at the right of the decimal point. The scale is set to 0 for integers or strings.*
attribute *precision* { xsd:positiveInteger } ,	*Maximum numbers of significant digits for the value, regardless of the position of the decimal point. For character strings, this represents the maximum length. The value of the precision attribute must be greater than the value of the scale attribute.*
attribute *primary* { "true" | "false" }	*If the value of this attribute is false, then this means that the associated data was computed by the device from other available primary data (hence it is not strictly necessary to save it). If the value of this attribute is true, or if this attribute is not present, then this means that the data is primary and that it must be saved.*
attribute *defaultvalue* { xsd:string }	*The value indicated by this attribute is used if the data element is present in the frame with no indication of value. This attribute thus provides a crude way of compressing the data flow for data elements which have a special value which occurs more often than other values. The value of this attribute is not saved in the NetCDF file, but it is used to save a default value when one was not specified.*
attribute *C_format* { xsd:string }	*A string providing the format that should be used by C applications to print values for this data element. For example, if you know a variable is only accurate to three significant digits, it would be appropriate to define the format attribute as "%.3g".*
attribute *quality* { name }	*Indicates the name of the data element associated with this data element and which represents the quality of values. If this attribute is missing, this means that no quality is associated with this data element.*
attribute *axis* { "X" | "Y" | "Z" | "T" }	*As specified by CF conventions, if this attribute is present, it indicates that the data element is an axis. For instance, a longitude is an X axis.*
attribute *coordinates* { xsd:string }	*As specified by CF conventions, this attribute indicates, when it is present, that the data element has coordinates which may be found in other data elements whose broadcastname are given, separated by spaces.*
attribute *comment* { xsd:string }	*Optional description of the data element, intended to be used for dynamic contextual help. This description should be kept short and simple.*
attribute *acquisitiontimedata* { name }	*Indicates the broadcastname of a data element which represents the acquisition time as measured by the device driver. If this attribute is missing, this means that no specific acquisition time is associated to the data and that the timestamp of the frame is a valid acquisition time. It is recommended to build the acquisition time data element by prefixing the group name by "ta".*
attribute *measuretimedata* { name }	*Indicates the broadcastname of a data element which represents the measure time of values as measured by the device and read by the device driver. For devices which transmit date/time of data in their message, the date/time of the measure is the one indicated in this message. For other devices, the algorithm used to compute the date/time of measure from the date/time of acquisition is left to the driver and is specified in the documentation of the driver.*
element **enum** {	
attribute *value* { xsd:string }	*If data has this value then it is mappable to the text below*
attribute *message* { xsd:string }	*Text that may be displayed in place of the "value"*
attribute *language* { xsd:string }	*Language associated with the message. This attribute allows the association of different messages to the same value, depending on the selected language. If no language is specified for a message, this means that the message is valid for every language.*
} *	
} +	
} +	
}	

```
messageid      =  attribute name { xsd:string }, version
frameid        =  attribute name { xsd:string }, version
dataid         =  attribute element_name { xsd:string } , version
description    =  attribute description { nonemptystring }
version        =  attribute major    { xsd:nonNegativeInteger } ,
                  attribute minor    { xsd:nonNegativeInteger }

name  = xsd:string { pattern = "[A-Za-z][0-9A-Za-z_]*" minLength = "1" maxLength =
"30"}
nonemptystring = xsd:string { minLength = "1" }
```

Formal specification in RelaxNG syntax of the structure of a description message

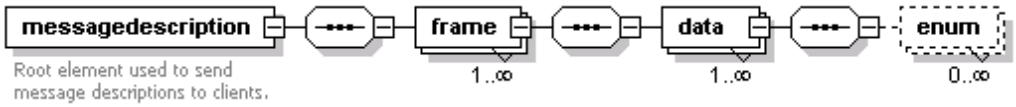

Figure 3 : Graphic presentation of the description of a XML description message

7.2 Example of a description message

```
<?xml version="1.0" encoding="UTF-8"?>
<messagedescription| xmlns="http://www.ifremer.fr/flotte/TECHSAS/2002"
xmlns:ts="http://www.ifremer.fr/flotte/TECHSAS/2002" name="bathysonde" major="1" minor="0">
    <frame name="bathysonde" major="1" minor="0" mandatory="true" sourcetype="batsnd"
               period="1" description="bathysonde specific data">
               <data broadcastname="depth" long_name="depth"
                         element_name="immersion" major="1" minor="0" units="m" type="float"
               cardinalitymin="0" cardinalitymax="1" precision="4" scale="2" C_format="%7.1g"
               valid_min="-20" valid_max="15000" missing_value="-100" axis="Z"
               positive="down" comment="bathysonde immersion"/>
               <data broadcastname="waterpressure" long_name="water pressure"
               element_name="water pressure" major="1" minor="0" units="bar" type="float"
               cardinalitymin="1" cardinalitymax="1" precision="1" scale="5" C_format="%8.1g"
               valid_min="0" valid_max="1500" missing_value="-1" positive="up"/>
               <data broadcastname="alt" long_name="altitude" element_name="altitude"
               major="1" minor="0" units="m" type="double" cardinalitymin="0"
               cardinalitymax="1" precision="7" scale="3" C_format="%7.1f" valid_min="-
               1000000" valid_max="30000000" missing_value="-10000000" axis="Z" positive="up"
               comment="altitude of the bathysonde above the bottom of the sea"/>
               <data broadcastname="nbbottles" long_name="number of bottles fired"
               element_name="nbbottles" major="1" minor="0" units="dimensionless" type="byte"
               cardinalitymin="0" cardinalitymax="1" precision="1" scale="0" valid_min="0"
               valid_max="100" missing_value="-1" positive="up"/>
               <data broadcastname="irrad" long_name="surface irradiance"
               element_name="surface irradiance" major="1" minor="0" units="dimensionless"
               type="float" cardinalitymin="0" cardinalitymax="1" precision="7" scale="3"
               C_format="%7.2f" valid_min="0" valid_max="999999" missing_value="-1"
               positive="up"/>
    </frame>
    <frame name="hydrology" major="1" minor="0" mandatory="true" sourcetype="batsnd"
                  period="1" description="generic hydrology data">
               <data broadcastname="temp" long_name="temperature" element_name="water
               temperature" major="1" minor="0" units="degree_Celsius" shortunits="°C"
               type="float" cardinalitymin="1" cardinalitymax="1" precision="7" scale="5"
               C_format="%7.3f" valid_min="-30" valid_max="120" missing_value="-300"
               positive="up" comment="temperature of water"/>
               <data broadcastname="conduct" long_name="conductivity" element_name="water
               conductivity" major="1" minor="0" units="mS/cm" type="float"
               cardinalitymin="1" cardinalitymax="1" precision="8" scale="6"
               C_format="%11.6g" valid_min="0" valid_max="10" missing_value="-1"
               positive="up" comment="conductivity of water measured inside the
               thermosalinometer"/>
               <data broadcastname="salinity" long_name="salinity" element_name="water
               salinity" major="1" minor="0" units="PSU" type="float" cardinalitymin="0"
               cardinalitymax="1" precision="6" scale="4" C_format="%8.4f" valid_min="0"
               valid_max="100" missing_value="-1" positive="up" comment="water salinity given
               by bathysonde"/>
               <data broadcastname="sndspeed" long_name="sound speed" element_name="water
               sound speed" major="1" minor="0" units="m/s" type="float" cardinalitymin="0"
               cardinalitymax="1" precision="6" scale="2" C_format="%6.1g" valid_min="1200"
```

```
                valid_max="1600" missing_value="0" comment="water sound speed derived from
                measures"/>
                <data broadcastname="density" long_name="density" element_name="water mass
                density" major="1" minor="0" units="kg/m^3" type="double" cardinalitymin="0"
                cardinalitymax="1" precision="6" scale="4" C_format="%9.6g" valid_min="900"
                valid_max="1200" missing_value="0" positive="up" comment="water density
                derived from measures"/>
    </frame>
</messagedescription>
```

7.3 Rules for building a description message

The best way to build a description message is to extract it from information found in the data dictionary. Description messages encoding is UTF-8.

Elements are emitted in the `http://www.ifremer.fr/flotte/TECHSAS/2002` XML name space.

7.4 Rules for decoding a description message

Any client program has to be able to deal with an XML message which does not conform to the specification, or with any non XML message;

Any client program can use information contained in the description message as it wants. For instance, a client which displays data will use the **long_name** and **C_format** attributes.

Chapter 8 NetCDF recording of XML broadcast messages

8.1 Version NetCDF

The NetCDF version 3.5 is used.

8.2 File name

Files are all written in the same directory. The file name has the following model : *YYYYMMJJHHMMSS-NAME-DEVICEID.TYPE*.

YYYY	year, 4 characters (ex :2002)
MM	month, 2 characters (from 01 to 12)
JJ	day, 2 characters (from 01 to 31)
HH	hour of file creation, 2 characters (between 00 and 23)
MM	minute of file creation, 2 characters (between 00 and 59)
SS	second of file creation, 2 characters (between 00 and 59)
NAME	name of XML source frame (cf. attribute **/messagedescription/frame/@name**)
DEVICEID	device ID (given by acquisition system) (cf. attribute **/configuration/device/@deviceid**)
TYPE	source type (cf. attribut **/messagedescription/frame/@sourcetype**)

The list of frames' names and the list of datatype is maintained in the data dictionary.

8.3 File contents

A NetCDF file contains data with frames with given **name**, **major**, **minor** attributes coming from an equipment with a given device ID and received by a record client between the file creation time and the file closing date.

As a matter of fact :

- All data from one equipment are saved in files which names match the **-DEVICEID.** expression.
- All data from one data source type are saved in files which names match the **.TYPE* expression.
- All data from one frame of one equipment are saved in files which names match the **-NAME-DEVICEID.** expression.
- The number of data in a file has no limit but the maximum size of a NetCDF file (2 Go).

8.3.1Global attributes

All created file has the following global attributes:

Title	Contains a short description of the file contents. This description is automatically generated by the record program and is based on its version.
History	Contains the record program name, its version number, the file creation date and the file closing date.
institution	Describes the place and name of the company or research center. This field is optional and its value depends on the record program configuration.
Source	Device name with "Acquisition of" prefix.
Comments	Option
Reference	Internet address of this document.
conventions	Contains the "CF-1.0" string.

creationtime (char)	File creation date. Its syntax is YYYY-MM-DDTHH:MM:SSZ. Milliseconds are not written.
frame_name (char)	name attribute of the frame.
frame_major (char)	major attribut of the frame.
frame_minor (char)	minor attribute of the frame.
frame_sourcetype (char)	sourcetype attribute of the frame.
frame_period (double)	period attribute of the frame, a `double` number expressed in seconds.

device_deviceid (char)	deviceid attribute of the equipment being recorded
device_devicename (char)	devicename attribute of the equipment being recorded
device_X (double)	x attribute (position in ship) of the equipment being recorded
device_Y (double)	y attribute (position in ship) of the equipment being recorded
device_Z (double)	z attribute (position in ship) of the equipment being recorded
device_position (char)	position attribute (position in ship, as a character string) of the equipment being recorded
device_firstusedate (char)	firstusedate attribute of the equipment being recorded
device_installdate (char)	installdate attribute of the equipment being recorded
device_latestcalibrationdate (char)	latestcalibrationdate attribute of the equipment being recorded
device_calibrationparameters (char)	calibrationparameters attribute of the equipment being recorded.
device_workingparameters (char)	workingparameters attribute of the equipment being recorded..
device_sourcetypelist (char)	sourcetypelist attribute of the equipment being recorded.
firstframetime (char)	First frame date of the file, written in the YYYY-MM-DDTHH:MM:SSZ syntax. Milliseconds are not written.
lastframetime (char)	Last frame date of the file, written in the YYYY-MM-DDTHH:MM:SSZ syntax. Milliseconds are not written. This value is initialized with 0000-00-00T00:00:00Z when creating the file and replaced by the correct value when closing the file. This system allows any further user to check if the file has been properly closed.

8.3.2 Dimensions

A NetCDF file contains one `time` dimension, which is the unique unlimited dimension of the file.

Other dimensions can be created, depending on the recorded variables (cf. p. 20).

8.3.3 Variables

Each data element which is described in a description message leads to the creation of one corresponding variable in the NetCDF file.

One `double time(time)` variable is created for each file. It has the following attributes :

Nom	valeur
long_name	acquisition time
Units	days since 1899-12-30 00:00:00 UTC
Calendar	gregorian
Axis	T

8.3.3.1 Variable name

One variable's name is given by the **broadcastname** attribute of the **<data>** corresponding element of the description message.

8.3.3.2 Variable type

The NetCDF type of a variable is given by the **type** attribute of the **<data>** corresponding element of the description message.

8.3.3.3 Variable dimensions

Each created variable has at least one `time` dimension.

Each variable of type `char` has a second dimension which name is the variable name, concatenated with `_LEN` and which value is the one of the **valid_max** attribute plus one (to take in account the NULL character at the end of character strings in C language).

Each variable which **<data>** corresponding element has a **cardinalitymax** attribute with a value strictly greater than 1 has a second dimension which name is the variable name, concatenated with `_LEN` and which value is the one of the **cardinalitymax**.

8.3.3.4 Variable attributes

All variable attributes are created and initialized from the attributes of the **<data>** element of the description message. When an attribute does not exist in the **<data>** element of the description message, it is not created in the NetCDF file.

The attributes order in the NetCDF file has no signification.

Name of XML attribute	Name of NetCDF attribute	Action
element_name	`element_name`	Copy from description message
major, **minor**	`element_version`	Concatenated **major** and **minor** attributes, and separated by a « . »
Comment	`comment`	Copy from description message
long_name	`long_name`	Copy from description message
Units	`units`	Copy from description message
Shortunits	`shortunits`	Copy from description message
valid_min	`valid_min`	Copy from description message
valid_max	`valid_max`	Copy from description message
	`valid_range`	Copy of **valid_min** and **valid_max** values, in this order, except for characters and string, for which the **valid_range** attribute is not created.
Precision	`precision`	Copy from description message
Scale	`scale`	Copy from description message
	`Add_offset`	fixed value 0
	`scale_factor`	fixed value 1
Positive	`positive`	Copy from description message
Primary	`primary`	Copy from description message
missing_value	`_FillValue`	Conversion of the attribute value in the variable type. Not created for a `char` type.
missing_value	`missing_value`	Valeur identique à `_FillValue`. Non créé dans les mêmes conditions
Quality	`quality`	Copy from description message
C_format	`C_format`	Copy from description message
acquisitiontimedata	`acquisitiontimedata`	Copy from description message
measuretimedata	`measuretimedata`	Copy from description message
Axis	`axis`	Copy from description message
Coordinates	`coordinates`	Copy from description message

8.4 File creation rules

One record program can create as many files as it needs. One recently opened file is considered as "opened" by the record program. It can add data in it as long as it receives data. A record program does never re-open a file it has closed. A closed file is considered as complete and no data has to be added by a later operation by the record program.

A file is closed for one of the following reasons :

- programmed recording time is elapsed,
- record program is stopped,
- limit size is reached,
- communication problem with the acquisition processes.

Given the maximum size of 2Go for NetCDF files and the fact that small files are easier to use and manipulate, the record program can close file at regular intervals.

8.5 Rules for transforming XML data message in stored data

The following actions are made when receiving an XML data message in order to store data :

- The message **deviceid** is saved in memory ;
- For each frame of the message, the **major** and **minor** are kept in memory. So is the frame name and the **acquisitiontime** value;
- If no NetCDF file is opened for the (equipment, frame) pair, a new NetCDF file is created ;
- The `time` dimension is incremented by 1
- The **acquisitiontime** value is saved at the `time(time)` position of the file ;
- For each frame element corresponding to an element in the description message, the corresponding value is stored at the time index of the variable, with the following tests:
  - If no value is given, the **defaultvalue** is given ;
  - If the variable type is a character string, the Null character terminating the string in C language is also saved in the NetCDF file ;
  - If the value is out of the **valid_min** and **valid_max** bounds, the program considers and behaves as if no data had been transmitted ;
  - If the variable is a vector, a counter of the element occurrences is set in order to save the values at their right place in the file ;

The time variable is not necessarily growing. It depends on the acquisition programs and communication between acquisition programs and record programs.

In order to follow the rules, the `_FillValue` attribute has to be properly set.