LASER INTERFEROMETER GRAVITATIONAL WAVE OBSERVATORY
- LIGO -
CALIFORNIA INSTITUTE OF TECHNOLOGY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Technical Note	LIGO-T980091-00	-E	98.10.08
Document	*Doc Number*	*Group*	*Date*

The LigoLW Format

An XML-based Language for Representing Scientific Data

Title

Kent Blackburn
Albert Lazzarini
Tom Prince
Roy Williams

Author[s]

This is an internal working note of the
LIGO Project

California Institute of Technology
LIGO Project - MS 18-33
Pasadena CA 91125
Phone +1.626.395.2966
Fax +1.626.304.9834
E-mail: info@ligo.caltech.edu

Massachusetts Institute of Technology
LIGO Project - MS 20B-145
Cambridge, MA 01239
Phone +1.617.253.4824
Fax +1.617.253.7014
E-mail: info@ligo.mit.edu

WWW: http://www.ligo.caltech.edu

1 Introduction

The objectives of the LigoLW language are to provide metadata and representation for collections of common scientific data objects (Array, Table, etc), but also keep the language simple, intuitive, and easy to create and use. Such objects often split naturally into metadata (also known as Header information, Parameter Files, Catalog cards) and data (a large binary file or list of numbers). The LigoLW file contains the metadata, and perhaps also the data, corresponding to a collection of data objects.

2 How do I use it?

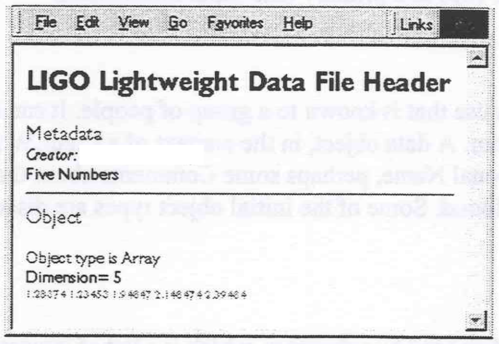
The syntax of LigoLW is based on XML (eXtensible Markup Language), now an industry standard for representing structured textual documents. XML combines the popularity of HTML in the wide Internet community with the battle-hardened power of SGML in the library community. Every LigoLW file is an XML file; some references to XML are at the end of this document. Here is a small example of a LigoLW file:

```
<?xml version="1.0"?>
<!DOCTYPE LigoLW SYSTEM "Ligolw.dtd">
<LigoW>
  <Metadata>
    <Creator>                </Creator>
    <Comment>Five Numbers</Comment>
  </Metadata>
  <Object>
    <Array> <Dimension> 5 </Dimension> </Array>
    <Data> 1.28374 1.23453 1.94847 2.148474 2.39484 </Data>
  </Object>
</LigoLW>
```

Figure 1: A sample LigoLW file that expresses an array of five numbers

The first line is like a "magic number" which must be the first line of any XML file. The second line says that this XML file is of a particular kind, a LigoLW file. The `<Metadata>` element expresses something about the objects that may be found below, and this is followed by a single `<Object>` of type `<Array>`. This Array has one dimension (there is only one `<Dimension>` element) and the contents are listed.

One of the advantages of basing the LigoLW format on a standard language such as XML, is that standard desktop tools can be used to view and edit the file. Figure 2 shows how this file looks when viewed with an XML-capable web



LIGO Lightweight Data File Header

Metadata
Creator:
Five Numbers

Object

Object type is Array
Dimension= 5
1.28374 1.23453 1.94847 2.148474 2.39484

Figure 2: The file of Figure 1 rendered by an XML-capable browser.

browser, such as Netscape 5.0 or Internet Explorer 5.0, (together with an appropriate style sheet, see section 4.3). Large numbers of XML tools are coming to market, for browsing, editing, sorting, and converting XML files. Some examples are shown for a more sophisticated LigoLW example later in this document.

The next question is how a program can read the data from the LigoLW file. There are three levels of answer to this:

- The user can delete all the non-data lines from the file and read it with `scanf()` in C or `READ` in Fortran.
- The user can select a parser in her favorite language from the list below, and use it to read the XML: perhaps she can start reading numbers as soon as a `<Data>` tag has appeared in the XML stream.
- The user can link code with not only a parser, but also a special library (not yet written) that converts the file to a C++ or Java object and returns it.

3 What is in a LigoLW File?

First, a little XML vocabulary. XML is a hierarchical languge of elements that may contain other elements. An ***element*** generally consists of a ***start tag***, a ***body***, and an ***end tag***, for example: `<Fruit>Banana</Fruit>`, where start and end tags are distinguished by the presence of a slash. An element may be ***empty***, meaning that there is only a single tag, with no body, for example `<EmptyElement/>`; note the position of the slash. Tags may contain ***attributes***, for example: `<Fruit color="yellow">`. For the format discussed in this document, there is a single `<LigoLW>` element which serves as the root of the hierarchy.

A LigoLW File consists of some metadata, together with a collection of data objects. The metadata defines a dictionary of name-value pairs, and the data objects define such things as arrays, tables, IGWD frames, and so on. The data corresponding to these objects may be stored in the LigoLW file itself, ot the data may be in a file, on the Internet, or on a tape from a suitably catalogued collection. The Object class mainly serves as a container class that provides the data portion of the object, together with optional `<Name>` and `<Comment>` elements.

3.1 Metadata

The metadata comes first in the file: there may be elements expressing the `<Creator>` and `<Date>` of the file, and there may be `<Comment>` elements. The rest of the metadata is a definition of a set of keyword-value pairs, whose meaning depends on the use of the file. An example might be:

```
<Key>
  <Name>Fruit_Mass</Name>
  <Unit>kg</Unit>
  <Value>0.387</Value>
</Key>
```

As may be obvious, the meaning here is "Fruit_mass = 0.387 kg", which is the kind of thing usually found in "parameter files" in scientific computing. Each <Key> element consists of mandatory `<Name>` and `<Value>` elements, so that users can associate parameters with their values. There may also be `<Comment>` and `<Unit>` elements. We expect programs to write these values into the LigoLW file, we expect people to edit them and attach comments to them with XML editors, and we expect programs to read these files as a primary data input mechanism.

3.2 Data Objects

A data object, in this context, supposedly has a meaning and a use that is known to a group of people. It can be represented — *serialized* — by ASCII text or by a binary stream. A data object, in the context of a LigoLW file, is something chosen from the above menu, together with an optional Name, perhaps some Comments about the object, and most importantly an indication of where the data may be found. Some of the initial object types are discussed in the next subsections.

3.2.1 Array

An array is a collection of numbers (or other primitive type) referenced by subscripts, which is a list of integers whose maximum values are given by the list of Dimensions of the Array. This definition is very close conceptually to a Fortran or C array.

3.2.2 Table

A table is an unordered set of *records*, each of the same form, where a record is an ordered list of values. The contents of a record are defined by column headings, each of which may have a unit and a type. This definition of a table is similar to the principle object that is found in a relational database.

3.2.3 IGWD Frame

We are implementing an XML definition of the metadata found in a IGWD Frame, the main data object of the LIGO and VIRGO gravitational-wave observatories.

3.2.4 MimeObject

We also intend to support a general object called **MimeObject**. This will be a way to connect to web servers that can deliver scientific data objects, (eg. images, web pages, sounds, etc) so that the LigoLW file can represent and collect more than the explicit classes mentioned above.

3.3 Where is the Data?

There is an assumption that each of the different kinds of data objects may be serialized as a sequence of integers, real numbers, etc, and that the metadata contained in the LigoLW file is sufficient to define this format. For example, in Figure 1, the dimension of the array is five, so that there are only five numbers in the Data section. The data may be explicitly included in the LigoLW file, as in the example above with the `<Data>` element, or it may be in another location, expressed by the `<Link>` element, or it may follow the closing `</LigoLW>` tag in the file.

3.3.5 `<Data>`: Embedded in the File

The data is explicitly embedded in the file, as in Figure 1. If the encoding (see next section) is not ASCII, or if the encoded data contains angle-brackets (< and >), then the LigoLW file is no longer a well-formed XML file and browsers and editors may not work properly.

3.3.6 `<Link>`: At Another Location

Often, scientific data is split between 'header' and 'data' sections. This model is supported by LigoLW through the `<Link>` tag, implying that the serialized data object is at a remote location; there is a URL-like syntax to express the external location, perhaps in a local file or on a numbered tape, or from a web or ftp server. There may also be hints about suitable timeout values and access privileges. While there may only be one `<Data>` element for a given object, there may be many `<Link>` elements, meaning that the same object can be mirrored in different places, and may be obtained from any of them.

3.3.7 `<Follows/>`: Data Follows

This empty element can mean one of two things, depending on whether or not there has been a previous `<Link>` tag in the LigoLW file.

- If no previous object had a `<Link>` tag, then the data follows immediately after the `</LigoLW>` end tag. There is a similar cautionary statement to the above, that the file is no longer a well-formed XML file, so many of the manipulation tools and parsers will need modification.
- If there has been a previous `<Link>` tag, then the data is assumed to follow immediately after the previous object. Thus if a collection of objects are all serialized in sequence in a file, then the LigoLW descriptor has a single `<Link>` element for the first object, and all subsequent objects have simple `<Follows/>` elements.

These elements are illustrated in Figure 3.

3.4 `<Encoding>`: Data Encoding

A part of both the syntax of both the Link and the Data elements is the Encoding of the data, which specifies how the numbers in the data are represented. Numbers are generally written as bytes in one of two ways: big-endian or little-endian; and binary data can be represented by an ASCII string in various ways, including uuencode or base64 encoding. The `<Encoding>` element will be more closely defined according to the dictates of experience.

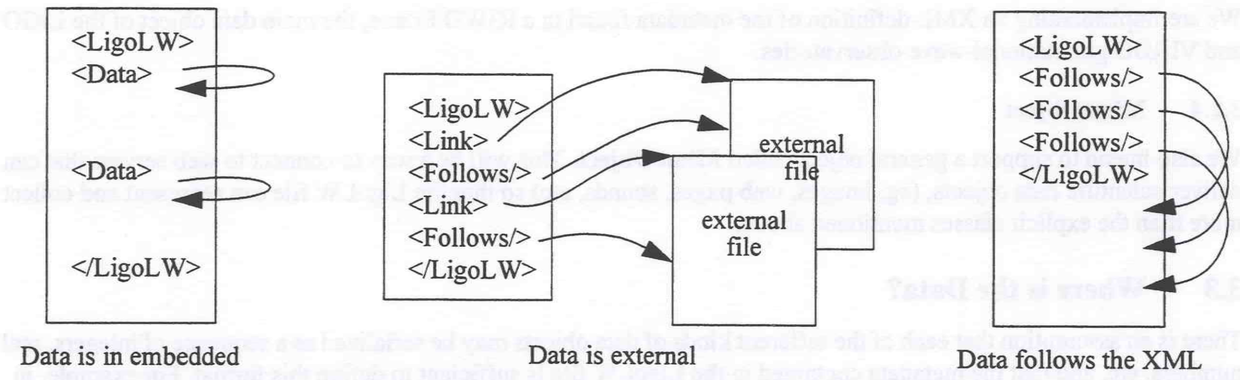


Figure 3: Illustration of the <Data>, <Link> and <Follows> elements.

4 An Example LigoLW File

To illustrate some of the capabilities of the LigoLW format, we have created a fairly comprehensive example, which consists of a one-dimensional array and a two-dimensional array, then a table, as shown in the following:

```
<?xml version="1.0"?>
<!DOCTYPE LigoLW SYSTEM "Ligolw.dtd">
<LigoLW>

<!-- First the Metadata ==================================== -->
  <Metadata>
    <Creator>Tom Prince</Creator>
    <Creator>Roy Williams</Creator>
    <Date>28 Sept 98</Date>
    <Comment>LIGO power spectrum of 32 magnetometers at 64 frequencies</Comment>
    <Key>
      <Name>LIGOType</Name>
      <Comment>The Ligo data type is defined here...</Comment>
      <Value>Power Spectrum</Value>
    </Key>
    <Key>
      <Name>StartDate</Name>
      <Comment>Can't remember exactly but this date is close!</Comment>
      <Value>03/21/97</Value>
    </Key>
    <Key>
      <Name>FreqSamp</Name>
      <Unit>Hz</Unit>
      <Comment>This is the sampling frequency</Comment>
      <Value>1024</Value>
    </Key>
  </Metadata>
<!-- Now for the Data objects ============================= -->
  <Object>
    <Name>Magnetometer</Name>
    <Array>
      <Dimension>64</Dimension>
      <Dimension>32</Dimension>
      <Type>double</Type>
    </Array>
```

```
<!-- This Array is at Cacr, Hanford, and on a tape -->
   <Link>
     <Encoding>bigendian</Encoding>
     <Timeout>600</Timeout>
     <Ref>file://hpss.cacr.caltech.edu/magval_09_25_97.bin</Ref>
   </Link>
   <Link>
     <Encoding>base64</Encoding>
     <Ref>file://hanford.ligo.caltech.edu/magval_09_25_97.bin</Ref>
   </Link>
   <Link>
     <Ref>tape://347846-6/756473</Ref>
   </Link>
 </Object>

 <Object>
   <Name>Magscale</Name>
   <Array><Dimension>32</Dimension></Array>
<!-- Embedded data -->
   <Data>
     1.28374 1.23453 1.94847 2.148474 2.39484 2.84746 3.10928 4.92827
     5.28374 5.23453 5.94847 6.148474 6.39484 6.84746 7.10928 8.92827
     9.28374 9.23453 9.94847 10.18474 10.3984 10.8446 11.1928 12.9827
     13.2874 13.2453 13.9847 14.18474 14.3984 14.8446 15.1928 16.9827
   </Data>
 </Object>

 <Object>
   <Name>Magoffset</Name>
   <Comment>This is the magnetic offset</Comment>
   <Array><Dimension>32</Dimension></Array>
<!-- Data follows from the end of the previous Object in the same stream -->
   <Follows/>
 </Object>

 <Object>
   <Name>Banana Species by Habitat</Name>
   <Table>
     Table specification TBD
   </Table>
 </Object>
</LigoLW>
```

In the following sections, various views of this file are shown from various tools.

4.1 Some Viewers and Editors

- Jeremie's Javascript Parser: Figure 4 shows a screen shot of a web page that does XML parsing on the client side. The client browser is sent a program in Javascript that does the actual parsing. For more information go here.
- Microsoft's XML notepad is shown in Figure 5. Here is a screen shot of the Microsoft product displaying a LigoLW file. This editor is in alpha release, available from here.
- XMLPro, as shown in Figure 6, is an editor for validated XML documents. Here is a screen shot of XMLPro when viewing the example LigoLW file. This product also validates the XML according to the DTD. The manufacturer is Vervet Logic: for more information, see the references.

4.2 Parsing with Expat in C

Of course the whole point of the LigoLW format is that it can be created, edited and used not only by humans, but also by machines. The following is a fragment of C-code that can be linked with the Expat parser. This is one of many parsers that are available, as listed at the end of this document. The user supplies three handler functions that handle the start and end of elements, and the text in between.

```
void startElement(void *userData, const char *name, const char **atts)
/* Called when a start tag is found, like this: <tag> */
{
    int i;
    char* s;
    int parity = 1;
    fprintf(fp, "<UL><LI><FONT COLOR=""#ff0000""><B>%s</B></FONT>\n", name);

/* The attribute list comes as a sequence key-value-key-value-... */
    while(s = *atts++){
        if(parity){
        fprintf(fp, "<LI><FONT COLOR=""#0000ff"">%s</FONT>\n", s);
        } else {
        fprintf(fp, "=<FONT COLOR=""#0000ff"">%s</FONT>\n", s);
        }
        parity = !parity;
    }
}

void endElement(void *userData, const char *name)
/* Called when an end tag is found, like this: </tag> */
{
    int i;
    fprintf(fp, "</UL>\n");
}

void textHandler(void *userData,  const char *s, int len)
/* Called when a chunk of text is found */
{
    int i;
    for(i=0; i<len; i++) putc(s[i], fp);
    fflush(fp);
}

int main()
{
  XML_Parser parser = XML_ParserCreate(NULL)
  XML_SetElementHandler(parser, startElement, endElement);
  XML_SetCharacterDataHandler(parser, textHandler);
/* fill buffer buf with XML */
  XML_Parse(parser, buf, sizeof(buf), finished_flag);
  XML_ParserFree(parser);
}
```

The results of running this C-code on the example file may be seen in Figure 7: the start tags are in bold red, and the body text in normal font. However this is not the real objective of parsing such a file; we intend the parser to be used when it is a computer that is to understand the content of the file, rather than a human.

We have used the parser by setting up handlers, or call-backs, and having the parser call these as necessary. Another way to use a parser is for it to create a tree structured object that the user can subsequently interrogate.

4.3 The XSL Style Sheet Language

An XML file such as our LigoLW format may be converted to a more human-readable form by creating a filter as above. But there is another solution that is gaining acceptance in the XML community; XSL (eXensible Style Language). This language expresses a set of rules and actions: when a certain pattern is found in the XML, then a given output sequence occurs. We illustrate this with the example LigoLW file, using an XSL file, part of which is shown below, in order to produce the HTML presentation that is shown in Figure 8. This conversion was done with the Microsoft `msxsl` tool.

```
<xsl>
  <rule> <!-- The root rule: what to do first-->
    <root/>
      <HTML><BODY background-color="white">
      <H2>LIGO Lightweight Data File Header</H2>
      <children/>
      </BODY></HTML>
    </rule>

<!-- The Metadata objects -->
  <rule> <!-- When you see a Metadata element, write the word Metadata in large red, then the content -->
    <target-element type="Metadata"/>
    <FONT color="red" size="+1">Metadata</FONT>
      <children/>
  </rule>

  <rule> <!-- When you see a Creator element, write the word Creator in italics, then the content -->
    <target-element type="Creator"/>
    <BR/><I>Creator: </I>
      <children/>
  </rule>

  <rule> <!-- When you see a Date element, write the word Date in italics, then the content -->
    <target-element type="Date"/>
    <BR/><I>Date: </I>
      <children/>
  </rule>
<!-- deleted -->
</xsl>
```

5 The Definition of a LigoLW file (DTD)

Finally we come to actually defining the LigoLW language as a subset of XML. We create a Document Type Definition (DTD), which defines the elements that may be used and the order in which they may be used The preliminary DTD for the LigoLW format is shown here:

```
<!-- ================================================================
Document Type Definition
Ligo Lightweight Data Format
Kent Blackburn, Albert Lazzarini, Tom Prince, Roy Williams
California Institute of Technology, September 1998
(preliminary)
================================================================= -->
<!ELEMENT LigoLW    (Metadata?,Object*)*>
<!-- ============================================================= -->
<!--Metadata may be a creator, date, a URL for more
  information, keywords, or some unstructured text
-->
<!ELEMENT Metadata   (Creator|Date|MoreInfoURL|Comment|Key)*>
<!ELEMENT Creator    (PCDATA)>
```

```
<!ELEMENT Date       (PCDATA)>
<!ELEMENT MoreInfoURL(PCDATA)>
<!ELEMENT Comment    (PCDATA)>
<!ELEMENT Key        (Name,Comment*,Unit?,Value)>
<!ELEMENT Name       (PCDATA)>
<!ELEMENT Unit       (PCDATA)>
<!ELEMENT Value      (PCDATA)>

<!-- ========================================================= -->
<!-- Objects have a Name and possibly some Comments,
then the subclass, then where to find the serialized object -->

<!ELEMENT Object (Name, Comment*,
                 (Array|Table|Frame|MimeObject),
                 (Data|Link|Follows)
)>
<!--========================= -->
<!--In the array object, the last dimension varies fastest -->
<!ELEMENT Array     (Unit?,Dimension*,Number?)>
<!ATTLIST Array
   Type (long|double|float|int|byte|char) "double">
<!ELEMENT Dimension  (PCDATA)>
<!ELEMENT Number     (PCDATA)>
<!--========================= -->
<!--Other DataObjects specifications TBD-->
<!ELEMENT Table      (PCDATA)?>
<!--========================= -->
<!ELEMENT Frame      (PCDATA)?>
<!--========================= -->
<!ELEMENT MimeObject (PCDATA)?>
<!--============================================================= -->
<!--Where is the Data?-->
<!ELEMENT Data       (PCDATA)?>
<!ELEMENT Link       (Encoding?|Timeout?|Ref)>
<!ELEMENT Encoding   (PCDATA)?>
<!ELEMENT Timeout    (PCDATA)?>
<!ELEMENT Ref        (PCDATA)?>
<!ELEMENT Follows    EMPTY>
<!--============================================================= -->
```

The DTD is the file referenced on line 2 of the LigoLW examples above. Let us explain just one line of the DTD in detail, line 8, which is:

```
<!ELEMENT LigoLW    (Metadata?,Object*)*>
```

This specifies that the overall container element, the `<LigoLW>` element, is made up from `<Metadata>` and `<Object>` elements. There may be zero or one `<Metadata>` elements (the question-mark notation) followed by some `<Object>` elements. The sequence of Metadata,Object,Object... may itself be repeated.

The rest of the DTD continues the definition. The word "PCDATA" implies 'parsed character data', essentially meaning plain text.

6 XML References

This list is not comprehensive. Currently (10/98) there are 38 books on XML at Amazon.com, and there are 230,000 matches at the Altavista search engine.

- This paper and much of the files and images can be found at the LIGO-CACR web site
  `http://www.cacr.caltech.edu/ligo`

6.1 Indexes and pointers

- Resources at XML.com
  `http://www.xml.com`
- XML Recommended Reading List (Adam Rifkin, Caltech)
  `http://www.cs.caltech.edu/~adam/local/xml.html`
- The World of XML Tools
  `http://www.computer.org/internet/v2n3/xml-tool.htm`

6.2 Articles

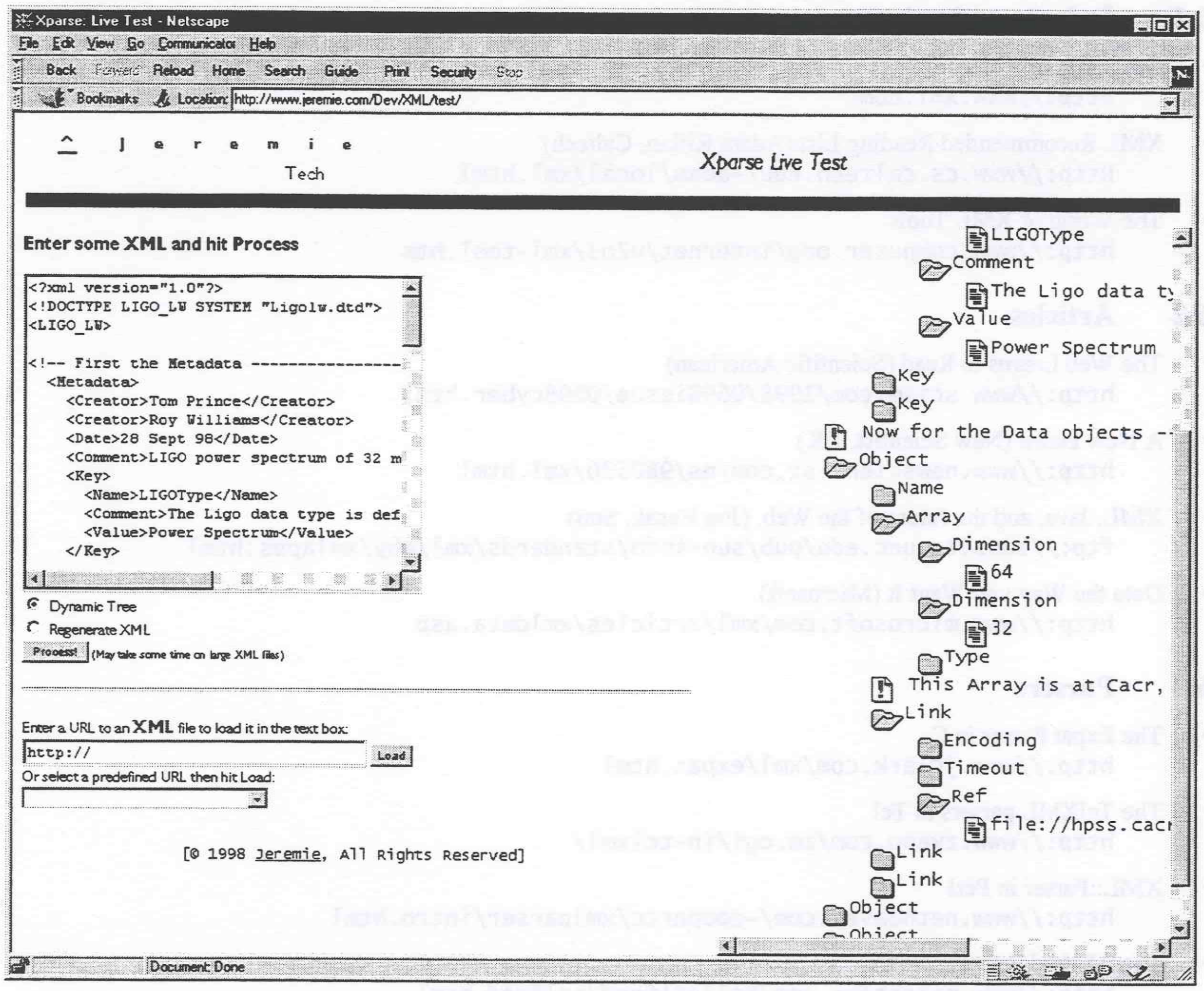
- The Web Learns to Read (Scientific American)
  `http://www.sciam.com/1998/0698issue/0698cyber.html`
- A New Dawn (New Scientist, UK)
  `http://www.newscientist.com/ns/980530/xml.html`
- XML, Java, and the future of the Web, (Jon Bosak, Sun)
  `ftp://sunsite.unc.edu/pub/sun-info/standards/xml/why/xmlapps.html`
- Data the Way you Want It (Microsoft)
  `http://www.microsoft.com/xml/articles/xmldata.asp`

6.3 Parsers

- The Expat Parser in C
  `http://www.jclark.com/xml/expat.html`
- The TclXML parsers in Tcl
  `http://www.zveno.com/zm.cgi/in-tclxml/`
- XML::Parser in Perl
  `http://www.netheaven.com/~coopercc/xmlparser/intro.html`
- The AElfred Parserin Java
  `http://www.microstar.com/Xml/AElfred/aelfred.html`
- The xmlproc Parser in Python
  `http://www.stud.ifi.uio.no/~larsga/download/python/xml/xmlproc.html`
- The Saxon Parser in Java
  `http://home.iclweb.com/icl2/mhkay/saxon.html`
- The Lark parser in Java
  `http://www.textuality.com/Lark`
- The Xparse parser in Javascript
  `http://www.jeremie.com/Dev/XML/index.phtml`

6.4 Browsers and Editors

- Jeremie's Xparse
  `http://www.jeremie.com/Dev/XML/index.phtml`
- Microsoft XMLNotepad
  `http://www.microsoft.com/xml/notepad/intro.asp`
- XMLPro (Vervet Logic)
  `http://www.vervet.com`


Xparse: Live Test - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://www.jeremie.com/Dev/XML/test/
Jeremie Tech
Xparse Live Test
Dynamic Tree
Regenerate XML
Process!
Load
Document: Done

Enter some XML and hit Process

```
<?xml version="1.0"?>
<!DOCTYPE LIGO_LW SYSTEM "Ligolw.dtd">
<LIGO_LW>

<!-- First the Metadata -----------------
  <Metadata>
    <Creator>Tom Prince</Creator>
    <Creator>Roy Williams</Creator>
    <Date>28 Sept 98</Date>
    <Comment>LIGO power spectrum of 32 m
    <Key>
      <Name>LIGOType</Name>
      <Comment>The Ligo data type is def
      <Value>Power Spectrum</Value>
    </Key>
```

(May take some time on large XML files)

Enter a URL to an XML file to load it in the text box:

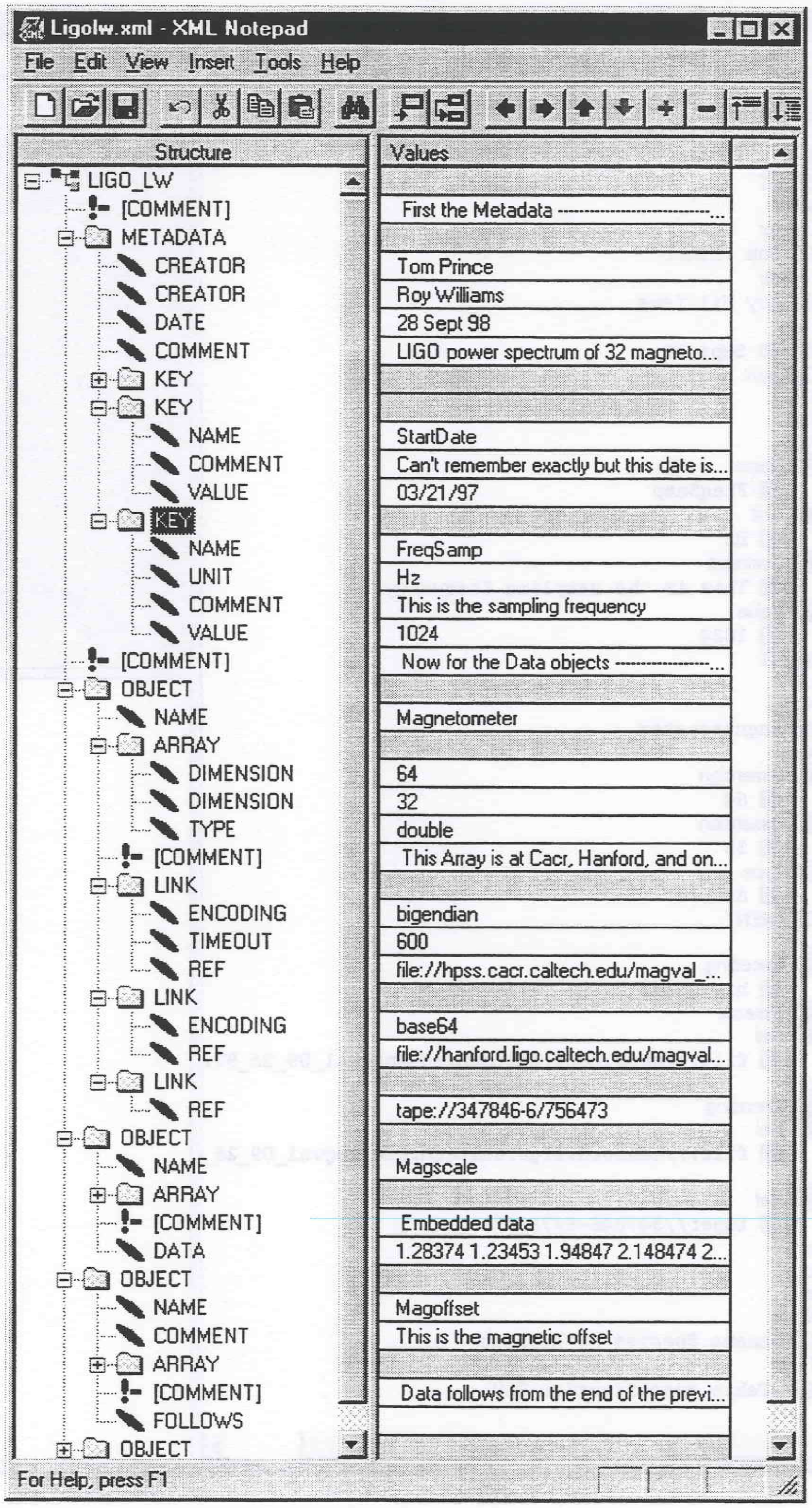
http://

Or select a predefined URL then hit Load:

[© 1998 Jeremie, All Rights Reserved]

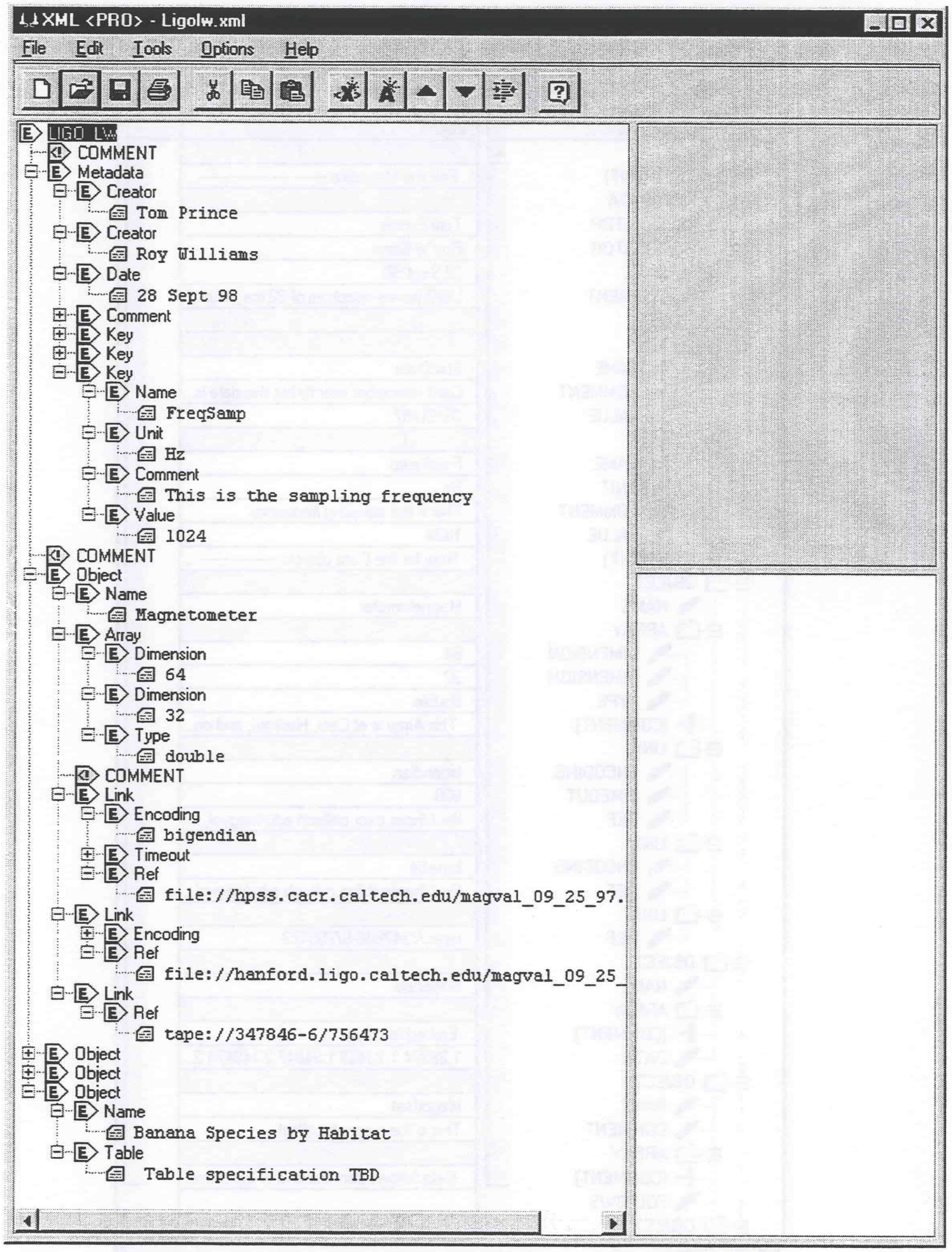
```
            LIGOType
        Comment
            The Ligo data ty
        Value
            Power Spectrum
    Key
    Key
 Now for the Data objects --
Object
    Name
    Array
        Dimension
            64
        Dimension
            32
        Type
    This Array is at Cacr,
    Link
        Encoding
        Timeout
        Ref
            file://hpss.cacr
    Link
    Link
Object
Object
```

Figure 4: The Xparse XML parser, written in Javascript (see below for URL).



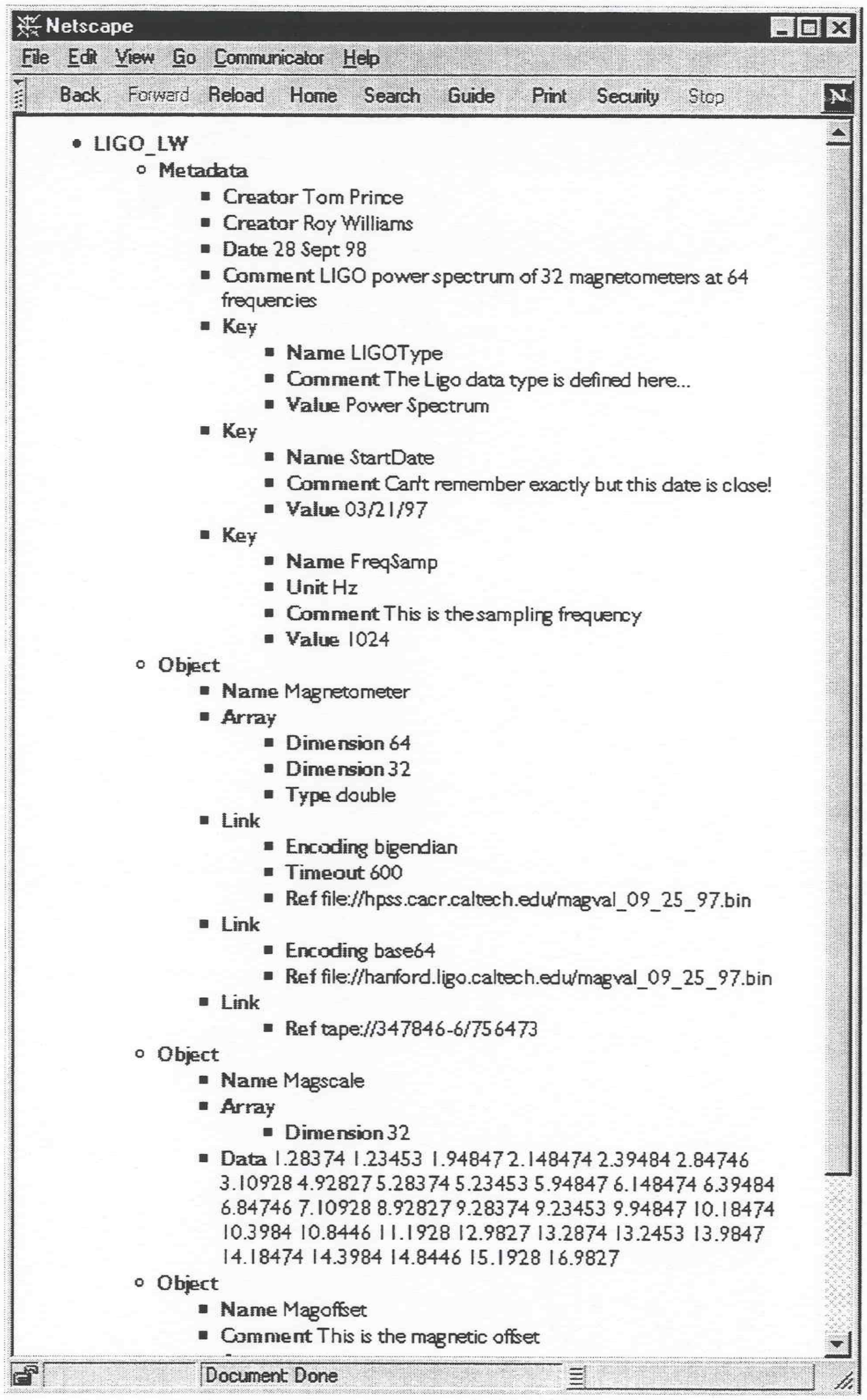
```
LIGO_LW
    [COMMENT]              First the Metadata ----------------------...
    METADATA
        CREATOR            Tom Prince
        CREATOR            Roy Williams
        DATE               28 Sept 98
        COMMENT            LIGO power spectrum of 32 magneto...
        KEY
        KEY
            NAME           StartDate
            COMMENT        Can't remember exactly but this date is...
            VALUE          03/21/97
        KEY
            NAME           FreqSamp
            UNIT           Hz
            COMMENT        This is the sampling frequency
            VALUE          1024
    [COMMENT]              Now for the Data objects ---------------...
    OBJECT
        NAME               Magnetometer
        ARRAY
            DIMENSION      64
            DIMENSION      32
            TYPE           double
        [COMMENT]          This Array is at Cacr, Hanford, and on...
        LINK
            ENCODING       bigendian
            TIMEOUT        600
            REF            file://hpss.cacr.caltech.edu/magval_...
        LINK
            ENCODING       base64
            REF            file://hanford.ligo.caltech.edu/magval...
        LINK
            REF            tape://347846-6/756473
    OBJECT
        NAME               Magscale
        ARRAY
        [COMMENT]          Embedded data
        DATA               1.28374 1.23453 1.94847 2.148474 2...
    OBJECT
        NAME               Magoffset
        COMMENT            This is the magnetic offset
        ARRAY
        [COMMENT]          Data follows from the end of the previ...
        FOLLOWS
    OBJECT
```

Figure 5: The alpha version of XMLNotepad from Microsoft. The URL is given below.



```
LIGO_LW
  COMMENT
  Metadata
    Creator
      Tom Prince
    Creator
      Roy Williams
    Date
      28 Sept 98
    Comment
    Key
    Key
    Key
      Name
        FreqSamp
      Unit
        Hz
      Comment
        This is the sampling frequency
      Value
        1024
  COMMENT
  Object
    Name
      Magnetometer
    Array
      Dimension
        64
      Dimension
        32
      Type
        double
    COMMENT
    Link
      Encoding
        bigendian
      Timeout
      Ref
        file://hpss.cacr.caltech.edu/magval_09_25_97.
    Link
      Encoding
      Ref
        file://hanford.ligo.caltech.edu/magval_09_25_
    Link
      Ref
        tape://347846-6/756473
  Object
  Object
  Object
    Name
      Banana Species by Habitat
    Table
      Table specification TBD
```

Figure 6: The XMLPro editor from Vervet Technologies. The URL is given below.


Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Document Done

- LIGO_LW
  - **Metadata**
    - **Creator** Tom Prince
    - **Creator** Roy Williams
    - **Date** 28 Sept 98
    - **Comment** LIGO power spectrum of 32 magnetometers at 64 frequencies
    - **Key**
      - **Name** LIGOType
      - **Comment** The Ligo data type is defined here...
      - **Value** Power Spectrum
    - **Key**
      - **Name** StartDate
      - **Comment** Can't remember exactly but this date is close!
      - **Value** 03/21/97
    - **Key**
      - **Name** FreqSamp
      - **Unit** Hz
      - **Comment** This is the sampling frequency
      - **Value** 1024
  - **Object**
    - **Name** Magnetometer
    - **Array**
      - **Dimension** 64
      - **Dimension** 32
      - **Type** double
    - **Link**
      - **Encoding** bigendian
      - **Timeout** 600
      - **Ref** file://hpss.cacr.caltech.edu/magval_09_25_97.bin
    - **Link**
      - **Encoding** base64
      - **Ref** file://hanford.ligo.caltech.edu/magval_09_25_97.bin
    - **Link**
      - **Ref** tape://347846-6/756473
  - **Object**
    - **Name** Magscale
    - **Array**
      - **Dimension** 32
    - **Data** 1.28374 1.23453 1.94847 2.148474 2.39484 2.84746 3.10928 4.92827 5.28374 5.23453 5.94847 6.148474 6.39484 6.84746 7.10928 8.92827 9.28374 9.23453 9.94847 10.18474 10.3984 10.8446 11.1928 12.9827 13.2874 13.2453 13.9847 14.18474 14.3984 14.8446 15.1928 16.9827
  - **Object**
    - **Name** Magoffset
    - **Comment** This is the magnetic offset

Figure 7: The result of running the expat code from section 4.2 and viewing the result in a Netscape browser.