DODS Data Standards Guide
Version 0.1

Tom Sgouros

August 30, 1999

© Copyright 1995-1999 by The University of Rhode Island and The Massachusetts Institute of Technology

Portions of this software were developed by the Graduate School of Oceanography (GSO) at the University of Rhode Island (URI) in collaboration with The Massachusetts Institute of Technology (MIT).

Access and use of this software shall impose the following obligations and understandings on the user. The user is granted the right, without any fee or cost, to use, copy, modify, alter, enhance and distribute this software, and any derivative works thereof, and its supporting documentation for any purpose whatsoever, provided that this entire notice appears in all copies of the software, derivative works and supporting documentation. Further, the user agrees to credit URI/MIT in any publications that result from the use of this software or in any product that includes this software. The names URI, MIT and/or GSO, however, may not be used in any advertising or publicity to endorse or promote any products or commercial entity unless specific written permission is obtained from URI/MIT. The user also understands that URI/MIT is not obligated to provide the user with any support, consulting, training or assistance of any kind with regard to the use, operation and performance of this software nor to provide the user with any updates, revisions, new versions or "bug fixes."

THIS SOFTWARE IS PROVIDED BY URI/MIT "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL URI/MIT BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTUOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE ACCESS, USE OR PERFORMANCE OF THIS SOFTWARE.

Preface

This document describes the various data standards that may be used by the Distributed Oceanographic Data System (DODS) data providers to facilitate the interoperability of DODS datasets. These standards are not essential to the operation of the DODS software, but are only meant as a guide to those providers for whom interoperability is an important consideration.

This document is relevant to version 3.0.0 and later of DODS.

0.1 Who is this Guide for?

This guide is designed to address the concerns of scientists and others who wish to use DODS to make their data available to one, two, or zillions of other scientists (and other others). Making data available using DODS is usually only half of the battle. It is often the case that for data to be actually useful to others, it must conform to certain data standards expected by other users or other users' software. This manual documents standards used and *recommended* (not required) by the DODS group.

This guide assumes a basic familiarity with the DODS software package. Refer to the *The DODS User Guide* for an overview of the package.

0.2 Organization of this Document

Chapter 1 provides an overview of the need for standards.

Chapter 2 describes one set of optional (and evolving) DODS data standards.

Chapter 3 shows an example: Making a simple dataset DODS compliant.

Appendix A on page 17 contains a list of the data variable names currently incorporated into the DODS meta-data standard.

0.3 Conventions

The typographic conventions shown in Table 1 are followed in this guide and all the other DODS documentation.

Table 1: Typographic Conventions

`Literal text`	Typed by the computer, or in a code listing.
User input	Type this precisely as written.
Variables	Used as a place holder for a user-specified or variable value. Choose an appropriate value and use that in place.
Button Text	Used to indicate text on a GUI button.
Menu Name	This is the name of a GUI menu.

When referring to a button in a menu, we will often use the notation: **Menu,Button**. For example, **Options,Colors,Foreground** would indicate the **Foreground** button in the Colors menu, selected under the Options menu.

Contents

List of Figures

List of Tables

1 DODS and Data Standards

Once upon a time there was DODS. And it was good. DODS solved many problems of dataset incompatibility by making remote datasets pretend to be in a format they were not in. This was tricky, and it was largely accomplished by hiding from the user the actual data storage format, and using many different converters to provide data in a standard transmission format.

However, data storage format isn't everything. Though the DODS solution is quite a useful one, it has its limitations. Two datasets can be incompatible even though they share an identical data storage format. For example, consider two sea surface temperature datasets stored in netCDF format files. One can store its temperature in degrees Celsius, and another can store temperature as raw data, in unscaled satellite counts, that have to be converted into temperature values meaningful to a human.

1.1 Levels of Compatibility

In discussions about whether two different datasets are compatible, we must distinguish between different levels of compatibility. What level is important depends to some extent on how you intend to use the data. To a human, for example, two datasets could be considered compatible if they are about the same data variable. Two temperature datasets that cover the same area might be considered quite compatible, if all one is doing is looking at two different color contour maps lying on a table, or side-by-side on a computer screen.

However if one is seeking datasets on which to execute various analyses and computations, a higher level of compatibility is required.

The issues of data compatibility can be separated into four categories:

- Data storage format
- Data attributes
- Data organization
- Server interface

1.1.1 Data Storage Format

Two datasets whose data are stored and retrieved with different data format software are incompatible. A dataset stored with netCDF software cannot be read by a program to read HDF files, and vice versa.

Data storage format was the first compatibility issue that DODS addressed, and it does so fairly successfully. Whatever storage format data are kept in, the DODS server translates them into a standard transmission format before sending them to the DODS client requesting that data. This translation happens file by file, however, so datasets that span multiple files may remain somewhat problematic, even though the file format itself is not an issue.

1.1.2 Data Attributes

Having solved the problem of data storage compatibility, DODS was faced with the problem of incompatible data attributes. That is, considering our two temperature datasets, one dataset could refer to its temperature data in an array called “T”, while another could refer to an array called “Temp”. To a human comparing the two files, the equation between the two might be obvious, but to a computer it might not. (Even to a human, the facts might not be as obvious as they seem. For example, “T” could also refer to “Time”)

Other data attributes that are important to be able to compute with a set of data include flags for missing data and unit and unit conversion information. The data standard outlined in Chapter 2 of this document is an attempt to address this issue.

1.1.3 Data Organization

The organization of a dataset can create another impediment to compatibility. Consider, for example, two datasets containing salinity measurements of the ocean at various different depths, and at different times. Assuming that there is some reason not to put all the information in a single file, each dataset must be divided among several files, and this is where certain incompatibilitles can sneak in.

One dataset could store its data in several files, each of which contains a three-dimensional array—corresponding to latitude, longitude, and depth—of salinity values, corresponding to a particular time. But a similar dataset might be arranged as arrays of latitude, longitude, and time, with several different files corrsponding to different depths. Both arrangements are, in fact, common among on-line datasets.

The DODS Catalog Server, under development (as of August 30, 1999), is intended to address the issue of data organization within a dataset, allowing a user to query differently organized datasets with the same syntax query.

1.1.4 Server Interface

The DODS software does create solutions to the compatibility issues outlined above. However, in doing so, it introduces a fourth compatibility issue, which is that of the server interface itself. A DODS server is programmed to respond to a set of server commands issued by a DODS client. These commands come in the form of URLs sent to the server.

Because the DODS project is a widely distributed project, and because the datasets available through DODS are—by design—not under central control, it is not always certain that two given DODS servers will be running software from the same software release. Similarly, since there are provisions for making local modifications to server behavior, it is also not certain that server functions available on one server are available on another, even when they are both running at the same software release.

Addressing these issues is an ongoing preoccupation of the DODS project and its (growing) user base. The DODS servers also contain provisions for making this information available to clients. See *The DODS User Guide* for more information.

2

The Emerging DODS Data Standards

Using DODS, a scientist or data center can make their data readily available to anyone who knows about it. But there is a difference between “readily available” and “readily usable.” Just because DODS can easily retrieve data doesn’t mean that it is easy to use.

The DODS project has found that the single biggest obstacle to ease of use of a dataset’s data is incompatibility of meta-data, or data attributes.[1] This incompatibility can range from the seemingly simple, such as different data names, to the far less tractable, such as incompatible time representations.

To address this problem, the DODS project has created the data attribute standard described in this chapter. This standard is *entirely optional*. A server can still serve data from a DODS dataset that does not conform to this standard, and a DODS client can still read that data. However, a dataset that conforms to this standard will be more easily read and processed by another user.

In a similar effort, attempting to assert some order over incompatible data attributes among netCDF files, the COARDS project (“Cooperative Ocean/Atmosphere Research Data Service”) has specified some standard attributes that shall be defined in a dataset. Note that this has little or nothing to do with netCDF itself. You can have a netCDF dataset that doesn’t comply with the COARDS standard. But it will be more useful to others if it does.

The DODS data standard described here is a superset of the COARDS standard. That is, datasets compliant with the DODS standard should also be COARDS-compliant.

Note that the DODS architecture allows data attributes to be added to any dataset *without* modifying the data files themselves. This makes the task of conforming to the data standard fairly simple, in most cases consisting of adding a few files to a directory.

[1] Please see the *The DODS User Guide* for a discussion of DODS treatment of attributes, or meta-data.

2.1 Essential and Optional Attributes

One of the motivating principles of the DODS project is to keep the burden on the data provider as light as possible. To further decrease the burden of making data available, the DODS data standard attributes are separated into two groups:

- Those that are essential to minimal data attribute interoperability, and
- Those that are very useful, but not essential.

The second group can be further split up among the more and less useful attributes.

The DODS project recognizes these as different "levels" of compatibility. For example, a level 0 DODS dataset is one that is served by DODS, but does not comply with the DODS data standards. A level 1 dataset conforms, but only contains the essential attributes. A level 2 datasets conforms and contains the essential and the level 2 attributes, and so on. The attributes described in the following sections are identified by their levels in this way.

2.1.1 Global and Variable Attributes

DODS attributes consist of sets of name-value pairs. To say that a data variable has attributes is the same as saying that there is a set of attributes (in the dataset's DAS) with the same name as this variable. The set can contain one or more name-value pairs, such as "units" and "degreesC" and "scale_factor" and "2".

Consider the dataset with a DDS like this:

```
Dataset {
  Int16 temp[time = 16][lat = 17][lon = 21];
  Float32 lat[lat = 17];
  Float32 lon[lon = 21];
  Float32 time[time = 16];
} test1;
```

The DAS for this dataset might look like this:

```
Attributes {
  temp {
      String units "degreesC";
      String long_name "Surface Temperature";
      String missing_value "-32767";
      String scale_factor "0.005";
      String DODS_Name "Temp";
  }
  lat {
      String units "degree North";
      String DODS_Name "Latitude";
  }
  lon {
      String units "degree East";
      String DODS_Name "Longitude";
  }
  time {
      String units "hours from base_time";
      String DODS_Name "Time";
  }
  DODS {
      String Convention "DODS", "COARDS";
  }
}
```

One of the attribute containers in this DAS does not correspond to any data variables. The attributes in this container are said to be "global" attributes, and modify the entire dataset. There is nothing special about the names; an attribute container can be called anything at all. However, the container called "DODS", if it exists, should contain the global attributes described in the DODS data standard, such as `Convention`, `Acknowledge`, and `history`.

2.2 Essential Attributes (Level 1)

The following attributes must appear in any DODS dataset for it to be considered level 1 compliant. These attributes must be defined for *each* data value in the dataset.

`units` A character array that specifies the units used for the variable's data. The units attribute should be formatted as per the recommendations in the Unidata `udunits` package.

`scale_factor` If present for a variable, the data are to be multiplied by this factor after the data are read by the application that accesses the data. One or both of `scale_factor` or `add_offset` must be present if the data are not stored in the specified units.

`add_offset` If present for a variable, this number is to be added to the data after it is read by the application that accesses the data. One or both of `scale_factor` or `add_offset` must be present if the data are not stored in the specified units. If both `scale_factor` and `add_offset` attributes are present, the data are first scaled before the offset is added.

`DODS_Name` Variable names should begin with a letter and be composed of letters, digits, and underscores. It is recommended that variable names be case-insensitive implying that the same case-insensitive name should not be used for multiple variables within a single file. Note that this is not an axis label (see the `long_name` attribute for that), but is a way to ensure that two datasets can be usefully compared with one another. The value for this must come from the list of DODS names. (See Appendix A on page 17 for the list, and for instructions for adding entries to the list that do not yet appear there.)

2.3 Optional Attributes

The optional attributes are divided into two different levels. This again allows a data provider some latitude in the level of compliance with the standard.

2.3.1 Level 2 Attributes

`missing_value` This is a conventional name for a missing value that will not be treated in any special way by the client application. Use this for missing elements in a time series, or for land areas in oceanic datasets, for example.

`long_name` A long descriptive name (title). This could be used for labelling plots, for example. If a variable has no `long_name` attribute assigned, the variable name should be used as a default.

`Convention` This global attribute is recommended to identify the dataset as conforming to the DODS data standard, identified here. The attribute should be a string with the value "DODS". Note that this is a global attribute, and should appear in the "DODS" attribute container, like this:

```
DODS {
  String Convention "DODS"; }
```

2.3.2 Level 3 Attributes

`Acknowledge` This string should contain an acknowledgement paragraph that can appear in papers that use the data from this dataset. For example: "The principal investigators in the production of these data are J.D. Elms, S.D. Woodruff, and S. Worley (http://www.cdc.noaa.gov/coads/participants.html)." and so on. This is a global attribute, and should appear in the "DODS" attribute container.

`history` Although not mandatory the `history` attribute is recommended to record the evolution of the data contained within a DODS data file. Applications which process this data can append their information to this attribute. This is a global attribute, and should appear in the "DODS" attribute container.

3

Adding Data Attributes

One of the original design goals of DODS was to permit data providers to use DODS with as little work as possible. A corollary to this requirement is that if ancillary information about a dataset is required, it must be very easy to provide. The DODS architecture allows a data provider to add attribute information to a dataset without modifying the data files at all. All that is required is a separate file, stored in the same directory as the data.

This chapter shows how you can easily add attribute information to an existing dataset to comply with the DODS data standard. We start by creating a small dataset and serving it with DODS Freeform ND Server. A similar example is presented in *The DODS Freeform ND Server Manual*, highlighting different aspects.

All the files used in this chapter are available on the DODS examples page.

3.1 A Worked Example

Consider the following table of numbers:

```
-47.303545 -176.161101  1.17125
-25.928001   -0.777265  2.07288
-28.286662   35.591879  2.36377
 12.588231  149.408117 -100.000
-63.223548   55.319598  0.04503
 54.118314 -136.940570  1.04085
-38.818812   91.411330  1.39978
-34.577065   30.172129  2.09096
 27.331551 -155.233735  2.30917
 11.624981 -113.660611  2.75036
```

This table represents a series of temperature measurements at ten different geographical locations. The latitudes and longitudes are given in decimal degrees, and the temperatures are given in units of ten degrees. That is, the temperature as recorded, times ten, equals the temperature in Celsius. There is one temperature measurement missing, and its place is marked with a value of `-100.0`.

(This dataset, and the ancillary files discussed in this chapter, are in the DODS examples directory, and is available at the DODS examples page, under `dasex.*`.)

The dataset shown here can be served as is, with the DODS Freeform ND Server, using the following format file (`dasex.fmt`):[1]

```
ASCII_data "lat/lon"
latitude 1 10 double 6
longitude 12 22 double 6
t 24 31 double 4
```

Put both of these files in your `htdocs` directory (refer to the documentation for your web server to locate this directory). After installing the DODS Freeform ND Server, you can issue data requests on this dataset. Use a web browser like netscape, and enter the following URL (Use your own server's name instead of `machine.edu`.):

```
http://machine.edu/cgi-bin/nph-ff/dasex.dat.dds
```

You should see something like this:

```
Dataset {
    Sequence {
        Float64 latitude;
        Float64 longitude;
        Float64 t;
    } lat/lon;
} dasex;
```

[1] See the *The DODS Freeform ND Server Manual* for more information about that software.

The is the DDS (Data Description Structure) of the dasex dataset. (See *The DODS User Guide* for more information about the DDS and what it means, as well as a description of what a Sequence is.) Instead of `.dds` at the end of the URL, you can use `.asc` to see the data itself. Try that.

To see data attributes, try the same URL, ending with `.das`:

```
http://machine.edu/cgi-bin/nph-ff/dasex.dat.das
```

You should see something like this:

```
Attributes {
 FF_GLOBAL {
   String Server "DODS FFND release 4.2.3";
   String Native_file " ...";
    }
}
```

The DAS is a list of the dataset's attributes. This dataset has two, and they are global ones. That is, they describe the entire dataset, and don't correspond to any of the data variables listed in the DDS. They are also in a container called `FF_GLOBAL`, which is another hint that they are global attributes. The `Server` string describes the actual DODS server that sent the data, and the `Native_file` string is a long one, containing information about the original file, and what it contained.

This is all very well, but what if someone wanted to know what units the `t` variable was in, or where the data came from, or what that value of -100 is supposed to imply, or even whether the variable represents temperature or time. The data themselves are silent on those issues, so either we have to provide additional documentation, or we have to introduce attributes for the data variables, to make the dataset self-documenting. We do that with an ancillary file of data attributes. Call this file `dasex.das`[2]:

[2] It is called dasex1.das in the DODS examples page. You will have to rename this file to make it work.

```
Attributes {
  t {
    String DODS_Name "Temperature";
    String units "DegreesC";
    Float64 missing_data -100.0;
    Float64 scale_factor 10.0;
  }
  latitude {
    String DODS_Name "Latitude";
    String units "degree_north";
  }
  longitude {
    String DODS_Name "Longitude";
    String units "degree_east";
  }
}
```

Now the DAS that is returned from the dataset looks like this:

```
Attributes {
    FF_GLOBAL {
        String Server "DODS FFND release 4.2.3";
        String Native_file "...";
    }
    temp {
        String DODS_Name "Temperature";
        String units "DegreesC";
        Float64 missing_data -100.0;
        Float64 scale_factor 10.0;
    }
    latitude {
        String DODS_Name "Latitude";
        String units "degree_north";
    }
    longitude {
        String DODS_Name "Longitude";
        String units "degree_east";
    }
}
```

Now you can see that `t` stands for "Temperature" and that the `-100` is missing data, and that all the measurements have to multiplied by 10 before they are in degrees Celsius.

Now that the data have these additional attributes, the dataset is compliant with level 1 of the DODS data standard. This can also be noted in the DAS, with the addition of another global attribute container called `DODS`. A file called `dasex2.das` contains the following:

```
DODS {
  String Conventions "DODS";
  String Acknowledge "Example dataset from the DODS documentation.";
  String history "Created for DODS standards book.";
}
```

Add this information to the bottom of the `dasex.das` file (*before* the last closing bracket), and now the dataset is compliant with level 3 of the DODS data standard.

A
DODS Names

Here is a list of the currently used DODS variable names. The most up-to-date list is available at http://unidata.ucar.edu/packages/dods/names.dat

Table A.1: Supported DODS Names

Name	Description	Units
`Sea`	Sea	Temp
`Sea`	Sea	Temp
`Sea`	Sea	Temp
`Sea`	Sea	Temp
`Sea`	Sea	Temp
`Sea`	Sea	Temp
`seconds since 1970`	seconds since 1970	
`degree`	degree	east
`degree`	degree	north

Index