DODS—Uniform Resource Locators

23 August 1996

Abstract

This document describes the syntax used by the Distributed Oceanographic Data System (DODS) to reference objects on the Internet. The syntax is based on the Uniform Resource Locator syntax used by the World Wide Web system designed at European Laboratory for Particle Physics. In addition to the syntax description, an overview of the use of resource locators by the main components of the system is given.

Note

This is a working document made available to solicit comments. To receive e-mail notice of new or significantly changed documents, send e-mail to majordomo@unidata.ucar.edu with 'subscribe dods' in the body of the message.

Hypertext references in the hardcopy version of this document are represented using footnotes that contain a Universal Resource Locator (URL) to the referenced document. To read the online documentation, open the URL 'http://www.unidata.ucar.edu/' with a World Wide Web browser.

This document is written for people interested in implementing software that will be part of, or work in parallel with, the Distributed Oceanographic Data System.

Contents

1 Introduction

The Distributed Oceanographic Data System (DODS) needs a syntax for names which describe the network location for different resources. There are many other systems which must solve a similar problem. Three different classes of solution to this problem have been implemented in existing systems: (1) The Network File System incorporates remote resources directly into the client computer's file system where they can be referenced as if they were local files, (2) `ftp` prompts the user for an Internet address and then directs all user actions to that address until it is told to disconnect and (3) World Wide Web (WWW) clients use a string, called a Uniform Resource Locator (URL), which combines the protocol, host and pathname. We choose the third approach after dismissing the first two as, respectively, too complex and too rudimentary.

A URL is a generalization of file naming concepts which is already in common use on the Internet. A URL extends the filename/pathname concept used to refer to files on a computer by adding an Internet address and protocol component. A URL specifies the computer and access method (protocol) in addition to the filename and thus can be used to refer to files anywhere on a network. While they are used in various forms by many different systems, URLs, as such, are most closely associated with the WWW project.

DODS chose the URL notation in part because it represents an evolving standard for resource specification on the Internet. Conforming, at least in spirit with this standard, puts DODS in a place to benefit from ongoing work to develop more general network resource names with minimal additional work. Because the existing definitions of resource locators (by both DODS and the WWW) do not adequately address some important issues, compliance with an existing *de facto* standard increases the chance that DODS will be able to take advantage of emerging solutions to those problems.

In addition, the syntax of DODS URLs must match that used by the WWW because the data delivery design uses the WWW HTTPD[1] server as its base-communication module. While previous designs for the data delivery system outlined the construction of special DODS-only data servers using RPC technology, the most recent design uses NCSA's HTTPD[2] for the communication component of all DODS data servers. Thus, each URL that refers to a data set must match the syntactical requirements of that software (see the documentation on HyperText Transfer Protocol[3]).

The Common Gateway Interface (CGI) mechanism of HTTPD is flexible enough to allow sophisticated specialization of the semantics of WWW URLs without requiring a change in URL syntax. DODS can further restrict the WWW syntax for its URLs through conventions supported by the DODS server tool kit and by forcing all access to take place via the HTTPD's CGI mechanism.

[1] http://hoohoo.ncsa.uiuc.edu/docs/Overview.html

[2] http://hoohoo.ncsa.uiuc.edu/docs/Overview.html

[3] http://www.w3.org/hypertext/WWW/Protocols/HTTP/HTTP2.html

Table 1: DODS URL Syntax

DODS URL	*Access* `://` *Host* `/` *CGI* `/` *Args*
Access	`http`
Host	*Domain name*
CGI	*CGI-directory* `/` *String*
Args	*Pathname*
	Pathname Constraint
	`(` *DODS URL* `)`
Constraint	`?` *Expression*

2 Structure and Interpretation of URLs

A formal grammar for WWW Uniform Resource Identifiers[4] (URIs) exists and provides a framework for extending the meanings of URLs beyond those used by the WWW project. In this section the syntax for DODS URLs is described (See Table 1) and the differences between the WWW and DODS URLs are also discussed. This syntax is less formal than that presented in the CERN documentation, however, DODS servers must be able to parse this syntax. In the case where a DODS server forms an interface between other DODS components and a foreign system, the DODS server must first parse this syntax and then translate the parsed expression into the foreign system's syntax.

2.1 Syntax of the DODS URL

Access The access component of a DODS URL is always the literal string `http`. The DODS data delivery architecture described in DODS—Data Delivery Architecture[5] uses the World Wide Web HTTPD[6] server developed by NCSA to transport data. Note that just about any WWW server will work for DODS as long as it supports the CGI mechanism.

Host The host component is an Internet-style host name that can be resolved to an Internet address.

CGI The CGI directory is the directory where CGI programs are kept. This varies from machine to machine but many UNIX computer use `cgi-bin`. In addition, it is possible to configure `httpd` to search several directories for CGI programs; the DODS dispatch CGI may reside in any of these directories. The string is the name of the DODS dispatch CGI. In the software distributed by URI/MIT this

[4] http://www.w3.org/hypertext/WWW/Addressing/URL/uri-spec.html

[5] http://www.unidata.ucar.edu/packages/dods/archive/design/data-delivery-arch/data-delivery-arch.html

[6] http://hoohoo.ncsa.uiuc.edu/docs/Overview.html

Table 2: Sample URLs which name resources; sometimes there is more than one acceptable way to name a single resource.

URL	References
`http://gso.uri.edu/cgi-bin/nc/data/fnoc1.nc`	the entire data set
`http://gso.uri.edu/cgi-bin/nc/data/fnoc1.nc?u`	`u` from the data set
`http://gso.uri.edu/cgi-bin/nc/data/fnoc1.nc/u`	`u` using the virtual file system syntax
`http://gso.uri.edu/cgi-bin/nc/data/fnoc1.nc?u<10`	all variables given that `u` is < 10 (Makes sense for sequence data only)
`http://lake.mit.edu/cgi-bin/jg/bloom/level1`	all variables in `level1`
`http://lake.mit.edu/cgi-bin/jg/bloom/level1/temp`	only `temp` in `level1`

is an abbreviation for the API (e.g., the dispatch CGI for NetCDF is `nc`). However, server implementors are free to choose any name they want. In addition, data providers may rename this script if they want to.

Args The URL components named *Args* contains a pathname used by the CGI to locate the data set. This may or may not be an actual path on the remote machine. In some cases the DODS server may be an interface to a DBMS. In this case the pathname is used (probably in conjunction with other information) by the CGI to formulate a query to the DBMS. The pathname may have a constraint expression appended to it. This is used either by the filter program or the CGI to limit the data access beyond the pathname. It provides additional flexibility in the way data is accessed through the server. The arguments may also be a second URL. In this case the *Access* and *Host* components of the outermost URL (called an *envelope*) specify the target process (server) which should receive the embedded DODS URL. The source process (client) does not interpret this embedded URL at all—only the target process attempts to evaluate it. Embedded URLs may be nested to any depth.

Constraint The constraint is a boolean expression used to constrain access to a variable. Different classes of variables can have different types of constraints. See DODS—Data Access Protocol[7] for a more complete discussion of constraints.

2.2 Characteristics of DODS and WWW URLs

DODS URLs are a proper subset of WWW URLs. URLs defined by the WWW project cover a wide variety of client-server systems while parts of DODS URLs are interpreted only to be DODS CGIs. WWW URLs include special cases for servers

[7] http://www.unidata.ucar.edu/packages/dods/archive/design/api/api.html

which support protocols such as `ftp`, `wais` and `gopher`. In these cases, the *Access* component of a WWW URL is used to indicate which protocol to use. When *clients* receive URLs, they use this information to select which protocol should be used to access the data referenced by the URL (i.e., the *Access* information is used to select the type of server with which the client will communicate). DODS URLs are only meaningful to CGI modules accessible over the network via HTTP servers. Thus, there will never be any variation in the *Access* component of a DODS URL. It will always be `http`.

In addition to a protocol restriction, DODS URLs always reference a CGI module on the server. This is necessary because all DODS data access takes place via one or more CGI modules. The CGI specification must follow the *Host* component of the URL.

While WWW URLs are the input to WWW browsers written with knowledge of URLs, and in many cases, the ability to parse parts of URLs, DODS URLs are passed to programs which nominally have no knowledge of URL notation. In DODS—Data Delivery Architecture[8] a data system architecture is presented in which user programs are re-linked with new implementations of data-access APIs. These *client libraries* understand and can manipulate URLs (and in particular, understand the particular restrictions DODS places on URLs) but the user programs do not. Some of the user programs were written before any of the URL software even existed.

A DODS URL references a data set, or portion of a data set. Accessing a DODS URL returns an experimental-type binary Multipurpose Internet Mail Extensions document. Such a document cannot be displayed by browsers nor does it contain URLs which link the document to other documents on the Internet. URLs that are part of the WWW system of HTTP servers reference documents which can be displayed by a large number of difference browsers. DODS URLs are not browsable at all, that is, they cannot meaningfully serve as input to one of the standard WWW browsers[9]. DODS uses URLs simply to reference data sets on the Internet and not for the more general purpose of linking a document within the larger context of all documents publicly accessible on the Internet.

There are two ways DODS CGI modules can interpret the part of the URL which the HTTP server passes to it; as a constraint expression or as an access to a virtual file system. This second method of interpretation is a specialization of the general behavior of HTTPD where the text following the CGI name is passed to the CGI module unprocessed. Accessing a data set as if it were a virtual file system means specifying variables within the data set as if they were files or files within directories subordinate to the CGI. For example in Table 2 the file name separator (`/`) is used to indicate that the within the named data set (the netCDF data set called *CDT*) only the variable `u1` is to be accessed. If the data set contained several levels of variables,

---

[8] http://www.unidata.ucar.edu/packages/dods/archive/design/data-delivery-arch/data-delivery-arch.html

[9] However, it is possible to add one of several suffixes to a DODS URL and get text information which a browser can display. See Section 3.2 for more information.

as in the examples for the three level JGOFS data set, then a specific level or variable within a level may be specified. However, the virtual file system syntax is not capable of specifying arbitrary set of variables from a data set unless those variables happen to be the sole members of a structure of sequence (See DODS—Data Access Protocol[10] for more information on data types DODS supports). The VFS syntax is a special case of constraint expressions. It is included in DODS because file systems are familiar tools to many users, much more so than boolean expressions, and because they are flexible enough for many user's demands.

The second syntax DODS supports for the portion of the URL past the CGI name is the constraint expression syntax. Constraint expressions are a way of passing, along with the URL, a set of restrictions to be applied to the data set when the client software (i.e., the surrogate library) accesses the data set. These constraints will be used during access to the data set to limit the variables and variable values extracted from it. The constraint expression, or portions of the constraint expression, will be evaluated *by the server* at the time of the data access. The syntax of the expression itself is dependent on the types of variables within the data set (See DODS—Data Access Protocol[11]).

3 Uses of URLs by DODS

This section briefly describes how URLs are used by the two major components of DODS: the Data Locator and the Data Delivery Mechanism.

3.1 URLs and the Data Locator

The Data Locator is used to find specific data sets within DODS. Users can query the Data Locator and expect that it will return a URL which can be used by other components in DODS to access that data set. These URLs may be used as supplied to user programs or they may be edited first. Because the URLs are ASCII text strings, no special software is required to edit or store them. In addition, DODS URLs may be obtained using other methods such as Mosaic, e-mail, or personal communications.

3.2 Data Delivery and URLs

The Data Delivery Mechanism consists of libraries with which users re-link their programs as well as stand alone utilities built at least partially from those libraries. URLs are used verbatim by the data delivery components of DODS. The URL is passed to the server or translator using the API *open* call (e.g., `ncopen()`). The client library stub is responsible for removing the envelope of the URL and sending either the path or embedded URL to the correct host/server or host/translator.

[10] http://www.unidata.ucar.edu/packages/dods/archive/design/api/api.html
[11] http://www.unidata.ucar.edu/packages/dods/archive/design/api/api.html

```
Dataset {
    Int32 u[time_a = 16][lat = 17][lon = 21];
    Int32 v[time_a = 16][lat = 17][lon = 21];
    Float64 lat[lat = 17];
    Float64 lon[lon = 21];
    Float64 time[time = 16];
} fnoc1;
```

Figure 1: The DDS of a dataset.

```
Dataset {
    Int32 u[time_a = 1][lat = 17][lon = 21];
} fnoc1;
```

Figure 2: The DDS of a dataset constrained by $u[2:2][1:17][1:21]$.

When a user wants to access data via a DODS server they must give to a client (i.e., a program linked with one of the DODS reimplemented API libraries) A URL which references that data. Users may choose to supply a constraint along with that URL effectively limiting the parts of the data set that the client can see. For example, suppose a data set exists which contains several arrays:

The user can refer to the entire dataset using a URL without a constraint expression; http://dods.gso.uri.edu/cgi-bin/nc/data/fnoc1.nc. If the user were to specify this, then the client-library will receive from the server a DDS like the one in Figure 1. However, if the user knows that they only want to work with a small part of the data set they can supply a constraint expression along with the URL. For example, suppose that the user only wants the latitude and longitude values for $time_a = 2$, and furthermore that they are only interested in the array u. They could supply the user program with the following URL: http://dods.gso.uri.edu/cgi-bin/nc/data/fnoc1.nc?u[2:2][1:17][1:21] The user program will receive DDS in Figure 2. For program which are designed to read the entire dataset without user interaction this is a powerful additional feature.

However, constraint expressions have a second use in DODS. They are used by the reimplemented APIs to extract specific parts of a data set when that is requested by the user program. Many APIs provide features which make it possible to write software which opens a data set, presents the user program with a collection of variables and then provides a way for the program to read one or more of those variables. In an API reimplemented for DODS, those calls must all be satisfied by information the API receives from a DODS server.

In order for the API to get information about the data set[12] the API must synthesize various URLs using the one given by the user as a base. For example, to get the DDS of the data set referenced by `http://dods.gso.uri.edu/cgi-bin/nc/data/fnoc1.nc` the reimplemented API must append the suffix `.dds` (e.g., `http://dods.gso.uri.edu/cgi-bin/nc/data/fnoc1.nc.dds`. Similarly, the DAS of that data set is obtained by appending `.das`.

Getting data is a bit more work than getting the DAS or the DDS. The request for a particular variable must be translated into a DODS constraint expression. This constraint expression is then appended to the URL given by the user and then the suffix `.dods` is appended to that. For example, suppose the user program makes an API function call requesting the value of the array u in the preceding figures. What the API is supposed to return is the values of the entire array u, but no other values. The reimplemented API would build the URL `http://dods.gso.uri.edu/cgi-bin/nc/data/fnoc1.nc?u.dods` where the $?u.dods$ specify that the variable u is to be the only variable in the projection and the `.dods` selects the data filter from the DODS server. For more information on the constraint expression syntax, see DODS—Data Access Protocol[13]; for more information on the DODS data servers, see DODS—Data Delivery Design[14].

4 Deficiencies of URLs

There are several features which URLs as defined by either DODS or the WWW lack. These include:

1. Poor support for version information
2. No support for location-independent naming
3. No formal support for quality rating of data sets
4. No support for binding extent information within the name-space (e.g., for how long will this data set exist)
5. No support for security information

While these are important issues, many are being investigated by the Internet Engineering Task Force (IETF) and the evolving Uniform Resource Name[15] (URN) standard. By following the de facto URL standard now, DODS is in a good position to benefit from the future standardization of URNs. Further, some of these deficiencies

[12] Metadata in one sense, although we shy away from that term since we have found that 'metadata' to one person is 'data' to another; the categorization often limits the usefulness of the underlying information.

[13] http://www.unidata.ucar.edu/packages/dods/archive/design/api/api.html

[14] http://www.unidata.ucar.edu/packages/dods/archive/design/data-delivery-design/data-delivery-design.html

[15] http://www.acl.lanl.gov/URI/archive/uri-archive.messages/1143.html

are not as severe as they might seem at first. For example, it is not clear that security information (providing access to a limited-access resource) belongs in a resource name. It might be more secure to design the resource so that it prompts specially for any security information required. Similarly, version and quality information may be incorporated informally to the existing URL syntax. Finally, a crude type of location independence may be achieved using domain name service aliases for important resources. These approaches are already in widespread use with the WWW.

Rather than form our own solutions for the deficiencies of URLs, DODS will wait until either the IETF or WWW community modifies URLs and/or adopts URNs or until continued development of DODS is significantly hampered by the problems.

5 Conclusion

In order to reference data sets on the Internet, DODS must use a naming convention which includes more information than the conventional pathname. Uniform Resource Locators (URL), as defined by the World Wide Web (WWW) project also serve that purpose. Because URLs are a de facto standard, adopting their syntax simplifies construction of DODS software, and eases integration of DODS with the emerging Internet information base.

DODS restricts the set of URLs it can use to a proper subset of the URLs acceptable to the WWW. This is done principally by stipulating that all data access takes place via one or more Common Gateway Interfaces (CGI) which extract data from the data set and return an experimental-type binary MIME document.

URLs, both as defined by the WWW project and in DODS's restricted sense, have a number of short comings. The URL notation does not carry information about the quality or longevity of the referenced resource. Furthermore, URLs are dependent on current naming conventions used on the Internet and thus lack 'location independence'. However, the focus of DODS is in providing network accessibility to scientific data and *not* solving the problems associated with URLs as currently used by much of the existing Internet. By using URLs, DODS stands to benefit from the ongoing efforts of groups that are addressing the known deficiencies of URLs.