NAME

vodata – query and access VO data services

SYNOPSIS

vodata [<*flags*>] [<*resource*> [[<*objname*> [<*sr*>]]]]

vodata [<*flags*>] [<*resource*> [[<*ra*> <*dec*> [<*sr*>]]]]

vodata [<*flags*>] [<*url*>]

OPTIONS

The *vodata* task accepts the following options:

–h,–-help
Print a help summary to the terminal and exit. No processing is done following this flag.

–v,–-verbose
Verbose output. The output will be more verbose than normal but exactly what is printed depends on whether other flags are enabled to changed the basic task behavior.

–-vverbose
Very-verbose output. Even more output.

The following flags control the major behavior of the task, i.e. the type
of output to present.

–a, --all Perform an action based on all available data. When used as part of a data query, this flag causes the <*resource*> argument to be used in a substring match of Directory *ShortName* or *Identifier* fields to create the actual list of resources to be queried. If the *ShortName* of a *TABULARSKY-SERVICE* from Vizier is given, the <*resource*> will typically expand to include all tables associated with the paper, and providing a means to access all of these tables from a single query.

–c, --count
Print only a count of the matching records found and do not save any results. The standard output for the task is to echo some of the input parameters and print a table of results showing progress and the number of matching records. If this flag is set, the output written to the screen will be the same, however the data will not be saved locally.

–g, --get
Get the data referenced by the results of a data query. This typically only applies to Simple Image Access service in which the result of a query include a column of "*access references*" to the actual data that must be resolved separately. Setting this flag will cause all data references to be resolved by the task once all of the data queries have been completed.

Access references are appended to a master "access list" as each query completes. In general the order in which these are retrieved cannot be guaranteed. Data downloads can be done in parallel by setting the number of concurrent max downloads using the *--maxdownloads=<N>* flag, the default is to download one file at a time. If this flag is followed with a comma-delimited list of numbers, only those rows in the result table will be accessed.

–m, --meta
Print only the column metadata for the named services. The output will be a list of the columns return by a data query to the service, but will not save the actual data. A default position and search size will be used for the query: In the case of Cone services a negative size is used, for SIAP services the FORMAT=METADATA flag is used in the query, and for tabular Vizier services the entire table is accessed. Compliant VO services will respond quickly with only the column metadata, tabular services may respond more slowly due to the need to transfer the data. Adding the *-v* or *--verbose-<N>* options will increment the VERBOSE level of services and may

return more metadata if available, to access these extra columns the same level of verbosity must be set during a data query.

The following flags specify data query options:

–b <bpass> or --bandpass=<bpass>
Constrain the query by bandpass. The argument following the flag must be one of the allowed bandpass specification string. Setting the flag will constrain any Directory search used to only those resources where the spectral coverage matches the given bandpass. Aliases for bandpasses are allowed, see below.

–i <file>
Specify a file containing the remaining positional command-line input. The command line is thought of as having the following components: the options beginning with a '-' character and their associated arguments, one or more <*resource*> names giving the service to invoke, an object name or position, and an optional query size. The '-i' flag allows everything except the options to be specified from a file (or the standard input if the '-' argument is used), creating in effect a means to interactively specify the e.g. resource/object without restarting the task, or to take these values from a file or input stream to create multiple independent queries. If either the resource or object name/position has already been specified they do not need to be specified again.

The format for the command file is the same as for the <*resource*>, <*objname*>, <*ra*> <*dec*>, <*url*>, or <*sr*> described below and as they would appear on the command line, all input lines are terminated with a newline, the file or input stream is terminated with an EOF. An example of how this may be used would be the using a command file such as:

```
2mass-psc  m31,m51  0.5
chandra  ngc4258,m51  0.25
```

The task will process this file as if the two lines had been invoked as separate commands. The advantage is that this input can be created dynamically by another task, and we can group resource and object lists into independent queries. See the *Examples* below for other uses.

–o <obj|file> or --object=<obj|file>
Specify the object name to use in a query. Object names are resolved automatically to J2000 equatorial coordinates. The argument to this flag may be the name of a single object, a comma-delimited list of object names, the name of a file containing object names, or the reserved value '-'.

The reserved value '-' tells the task to take this information from the standard input, processing doesn't begin until the object list has been fully read.

–p <pos|file> or --pos=<pos|file>
Specify the position to use in a query. Positions are composed of two values assumed to be equatorial J2000 coordinates. Values specified as a floating-point decimal are assumed to be in units of degrees, sexagesimal values may also be used and are assumed to be equatorial RA and Dec. If the <*pos*> arg is used only one set of coordinates may be given on the command-line and must be delimited by a comma, however the argument may also be the name of a file containing coordinates to be processed, or the reserved value '-'.

The reserved value '-' tells the task to take this information from the standard input, processing doesn't begin until the position list has been fully read.

–r <radius> or --sr=<radius>[<units>]
Set the search radius. The default search size is 0.1 degrees unless specified on the command-line and argument are assumed to be in degrees, setting the size in other units is permitted using the *-sr* flag. To specify the <units> for the *--sr* flag, the argument should be suffixed by an 's' to

specify arcseconds, an 'm' to set arcminutes, and 'd' to set the size in degrees. By default, all queries will be done using the same search size. Variable search sizes accomplished using the '-i' flag described above.

–s <service> or --svc=<service>
Specify the service or url to invoke. In most cases the service, i.e. the <*resource*> argument will be taken from the commandline based on it's position. The exception is when the user want to specify a service URL directly (e.g. to test a local service) because it isn't known to the Directory, or to use the reserved values '-' or 'any'. Use of '-' tells the task to read the service list from the standard input; use of the word 'any' is a means to telling the task to dynamically create the resource list from other options (e.g. "any image service" by using the

–t <type> or --type=<type>
Constrain the query by service type. The list of allowed service types is given below. The actual string used in a Directory resource record may be used if known, otherwise common use is to specify 'image' to access Simple Image Access (SIAP) services, 'catalog' for Cone searches, or 'table' for Vizier tabular data.

The following flags are specific to the writing of HTML or KML files:

−-webnoborder
Disable the shaded border drawn around an HTML table.

−-webnocolor
Disable the coloring for an HTML table.

−-webnoheader
Disable the HTML page header written to the output file.

−-kmlmax=<N>
Specify the max number of placemarks to write. The default is 50, ordering is not guaranteed. Setting the sampling will automatically increase the maximum number of results returned.

−-kmlsample=<N>
Specify the sampling of the result to be every <*N*> rows. The default is to write all rows to the output file. If set, this value will be used as a multiplier for the max number of placemarks automatically.

−-kmlgroup=object
Group the results of a multi-resource/multi-object query into a single hierarchical KML file grouped by the object or position index (default);

−-kmlgroup=service
Group the results of a multi-resource/multi-object query into a single hierarchical KML file grouped by the service name.

−-kmlgroup=both
Groups the results of a multi-resource/multi-object query into a single hierarchical KML file. The two top-level folders will be 'By Source' and

−-kmlnolabel
Disable the labelling of placemarks. By default, the ID_MAIN ucd for each point will be used as a label.

−-kmlnoregion
Disable the drawing of the region bounding box in a KML file.

−-kmlnoverbose
Disable the writing of verbose information to the KML placemarks. By default, each placemark will contain all information from that result.

Input Options:

--cols=*col_str*
: Use columns specified in *col_str* to read the *ra*, *dec* and *id* values respectively. *col_str* is a comma-delimited list where the *id* column is optional and will not be used if not present as the third element in the list. Other columns may be given as a single integer or as a range of the form *start-end* indicating the values in the *start* thru *end* columns should be combined into a single value.

–d, --delim=*delim*
: Use the *delim* as the input table delimiter. By default, a space, tab, comma, vertical bar ('|'), or semicolon may be used as a delimiter for the input table. If no explicit delimiter is specified, the first occurance of any one of these will be used. The reserved words *comma*, *space*, *tab*, or *bar* may be used in place of a specific character.

<DT><B>--ecols=</B><I>col_str</I>

--ecols=*col_str*
: Use the explicit columns specified in *col_str* in the input table. This option should only be used with formatted text tables where the desired values will always be in the same columns of the file. Note that 'column' in this case refers to a specific character column in a text file. Columns may be a single integer or a range, and is a comma-delimited list as with the *--cols* option.

–f, --force
: Force the input table to be used even in the number of columns varies on each line. The assumption here is that any variation (e.g empty columns) occurs after the *ra*, *dec* and *id* columns in the table.

--hskip=*<N>*
: Skip *<N>* header lines in the input file. This option is only needed when the lines to be skipped do not begin with the normal '#' comment character.

--nlines=*<N>*
: Use only *<N>* lines of the input table.

--sample=*<N>*
: Sample the table every *<N>* lines. Setting the sample will not affect the *nlines* used.

Output Options:

–1,--one
: Save the results into a single file regardless of format. This option will be set automatically if the output is being written to the standard output. If the output format is something other than KML or XML, all results will be concatenated into individual files of the form "<svc>_<pid>.<extn>" so that each file will contain the object results from each service where the columns will be the same.

–A,--ascii
: Save the results as a whitespace delimited ascii table. If an output file is created it will have a ".txt" extension appended automatically.

–C,--csv
: Save the results as a comma-separated-value (CSV) table. If an output file is created it will have a ".csv" extension appended automatically.

–H,--html
: Save the results as an HTML table. If an output file is created it will have a ".html" extension appended automatically. See above for the *--webnoheader* option that can be used to disable the HTML page header.

–I,--inventory
Query the *Inventory Service* rather than the data services directly. This will return simply a count of the results found, but when presented with a table of resources and sources can be used to do a simple crossmatch of the sources found in the catalogs available through the service.

–K,--KML
Save the results as a Google Earth/Sky KML placemark file. If an output file is created it will have a ".kml" extension appended automatically. See above for additional options that control the content of the file.

–R, –V or --raw, --votable
Save the results as a raw VOTable. If an output file is created it will have a ".vot" extension appended automatically.

–T,--tsv Save the results as a tab-separated-value (TSV) table. If an output file is created it will have a ".tsv" extension appended automatically.

–O <root> or --output=<root>
Set the root of the output name. The reserved value '-' tells the task to write to the standard output.

–X,--xml
Save the results wrapped XML file of the raw VOTable results. If an output file is created it will have a ".xml" extension appended automatically. The XML document will gather all the individual VOTable result files to a single XML document, where each entry is wrapped by the element *<VOTABLE_ENTRY>*. There will be three attributes: *svc* will be the data service name, *obj* will be the object name (if supplied), and the *index* attribute giving an index into the results. This index list is created by looping over each service, and for each service, looping over the object/position list.

–e,--extract
Extract positional and access information to extra output files. By default both files will be written, using *--extract=pos* will write only the positional information file, using *--extract=urls* will write the access URLs only. Access URLs are written one-per-line to a file with the same root name as the main output but with a ".urls" extension; Positional information is written to a file with a ".pos" extension and will contain three columns made up of the identifier (the column with the *ID_MAIN* ucd), RA and Dec (the *POS_EQ_RA_MAIN* and *POS_EQ_DEC_MAIN* ucd columns respectively). If these ucds appear more than once in a table, the first occurrance will be used.

Additionally, the *--extract=headers* and *--extract=kml* flags may be to to specify the HTML and KML output be written to files with ".html" and ".kml" extensions respectively. The *--extract=KML* flag will cause multi-resource and/or multi-object queries to be collected into a single KML file. The format-specific *--kml<opt>* and *--web<opt>* flags will apply to these files. A *--extract=xml* flag will force the output format to be raw VOTable and gather the results to a single XML document (see the *-X* option).

Note that the URLs file can be used to later access the data (perhaps after sub-selecting from the table based on some criteria) by calling the task again using the filename as the only argument.

–n,--nosave
If enabled, this flag tells the task not to save results to local disk. Status and result information will continue to be printed to the screen, but no data are saved to disk.

–q,--quiet
Quiet mode. Suppress any extraneous output and warning messages.

–u,--url Force the specified URL to be downloaded.

DAL2 Query Options:

--band=*band_string*
The spectral bandpass is given in range-list format. For a numerical bandpass the units are wavelength in vacuum in units of meters. The spectral rest frame may optionally be qualified as either *source* or *observer*, specified as a range-list qualifier. Bandpass names are often not useful for spectra (they are probably more useful for image or time series data) but there are cases where they are useful for spectra, for example for a velocity spectrum of a specific emission line.

--time=*time_string*
The time coverage (epoch) specified in range-list form as defined in section 8.7.2, in ISO 8601 format. If the time system used is not specified UTC is assumed. The value specified may be a single value or an open or closed range. If a single value is specified it matches any spectrum for which the time coverage includes the specified value. If a two valued range is given, a dataset matches if any portion of it overlaps the given temporal region.

DESCRIPTION

The **vodata** task allows a user to query and access VO data for multiple resources and objects from a desktop or scripting environment. By design, the task interface is meant to provide the following features:

- Resources (i.e. data services) may be referred to using a more familiar *ShortName* designation, or an IVO identifier, either of which will be resolved to a specific *ServiceURL* internally using the Directory.
- Object names may be used to specify the location of a data query, the position will be resolved internally using the *Sesame* web service.
- Output files may be created in a variety of common formats easily manipulated with other desktop tools.
- Multiple resources and objects shall be queried in parallel when possible to optimize the task.
- Data referenced in a query response should be accessible by the task automatically.
- The command-line interface should be as friendly and as flexible as possible to allow the task to be used in multiple ways.

The task should quickly become familiar to users and is meant operate in concert with the **vodirectory** and **vosesame** tasks to allow novice users to begin to explore for data resources to be used in the final query. Some of the flexibility of the task is shown in the Examples section below. Major concepts of the task are detailed below as well.

Argument Parsing

The meaning of the various command-line arguments is detailed below:

<resource>
The *ShortName* or *Identifier* of a data resource to be queried, a comma-delimited list of either, or the name of a file containing either. These names will be resolved to a data service URL using the Directory. The *-s* option may be used to specify a non-registered *ServiceURL* that the task may use, however the *-t* option is then also required to specify the type of service.

<objname>
The name of an object, a comma-delimited list of object names, or the name of a file containing object names. The coordinates of each object will be resolved to a position prior to processing using the *Sesame* name resolver service. An error will be returned if an object name cannot be

resolved, and that object will be skipped.

<ra> <dec>
The J2000 equatorial RA and Dec position to the searched. Values given as floating point values are assumed to be in decimal degrees, sexagesimal values are assumed to be equatorial RA/Dec positions. Sexagesimal values may be of the form *hh:mm:ss.s* or *hh:mm.m* for RA, or *dd:mm:ss.s* or *dd:mm.m* for Dec. Only one coordinate pair may be specified on the commandline.

<sr> The search size for the data query specified in decimal degrees. The default size of 0.1 degrees will be used if this is not specified on the command line. The *-rs* and *-rm* options may be used specify the size in arc seconds and minutes respectively. The *-i* option may be used to specify command-line input options, where each command-line can include a different value for the search size, otherwise only one value is allowed.

<url> A single URL, or the name of a file containing URLs listed one per line.

Multi-Thread and Multi-Process Data Querying

All data queries require at least one *resource* and one *source* to be successful. The *resource* defines a specific data service to be queried, and the *source* is either an explicit position on the sky or the name of an object that can be resolved to a position. Additional parameters to the query are used to specify other options, but in essence each data query is translated to a single URL that must be accessed by the client task. In a complex query, lists of resource and/or objects create a potentially large matrix of queries that must be made (i.e. *N-services* by *N-objects* in total). Because a large fraction of the time spent in waiting for a query to finish is in waiting for the server to respond, we are able to run multiple queries simultaneously without saturating our network bandwidth in most cases.

The **vodata** task will parallelize the list of services to be queried by running a separate processing *thread* (i.e. a lightweight process running in parallel within the main application) for each of the services to be called. This allows queries to different servers to be run in parallel, and since these servers will often reside on multiple machines the client won't impact any one data provider too badly. In addition, the list of objects to be queried at each service will be broken up into multiple child processes and called simultaneously. This allows, for example, 10 objects to be queried from 3 services (a total of 30 queries) simultaneously.

The *--maxthreads=<N>mt* option can be used to set the max number of threads to be created for processing the resource list (the default is 20). If the resource list is larger than this value, the list will be processed with no more than the max number running at any one time until all resources have been queried. Similarly, the *--maxprocs=<N>* option can be used to set the number of child processes to be created to process the object list (the default is 10). When setting these values it is important to remember that the total number of *potential* processes running on your machine will be the product of these to values. The default values were empirically found to work reasonably well on most modern machines.

Additionally, it is worth considering the potential strain that can be put on data providers' machines before changing these settings. The large majority of Cone services for example come from a single server at HEASARC and overloading the server with hundreds of requests to multiple resources it provides may result in a failed request and what would appear to be no data. One should consider using the *-i* flag as a means to query a large object list against a resource list such that only the object processing is parallelized and the server load is minimized (See the example below).

Output Filename Generation

The *-O* option may be used to specify the root part of output files created by a data query. However, to guarantee that a multi-service, and/or multi-object query doesn't overwrite a single output file, the filename root will also include the *pid* (process ID) of the task that created it. For a single service and object query no *pid* will be used as part of the filename. This scheme guarantees unique output files across the various processing scenarios, with similar root names for multiple files associated with a specific query.

Output tables may be created in a number of formats and will likewise have extensions indicating the table type. The *-e/--extract* option may create additional files for each query, and the *-g/--get* option to access data will similarly create additional files. The structure of an output filename is:

<root>[_<pid>].<extn>

The meaning of *<pid>* and *<extn>* have been discussed above. If the *-O* option was set then the *<root>* part of the name will simply be the argument given to set the root name. Otherwise, the *<root>* element will be of the form:

<svc>_<type>_<objname>
<svc>_<type>_<index>

The *<svc>* is derived from the service name used, the *<type>* is a single-character code to indicate the type of service used ('I' for image, object name or the index in a list of positions of no object was specified.

Verbosity

The *-v* and *-vv* options serve a dual purpose: within the task they set the level of output verbosity in terms of what is reported during processing (Similarly, the *-q* option can be used to turn off output reporting entirely). These flags will however also increase the value of the **VERBOSE** parameter sent to services during a data query. The default value is at least 1, with the highest level being 3. Using the *-v* flag sets **VERBOSE=2** and *-vv* sets **VERBOSE=3**.

The VERBOSE level can be important in accessing result columns that may only be returned at the highest level. When using the *-m* (or *--meta*) flag to print the column metadata, the verbose options will also affect the results and it is important that the same verbosity be set when doing the actual data query and access.

Bandpass and Service Type Aliases

The type constraint (*-t* or *--type*) accepts only the following arguments:

catalog	Cone search services
image	Simple Image Access services
spectra	Simple Spectral Access services
table	Vizier services
<literal>	ResourceType from Directory record

The bandpass constraint (*-b* or *--bandpass*) accepts only the following arguments:

Radio	Millimeter	Infrared (IR)
Optical	Ultraviolet (UV)	X-Ray (xray)
Gamma-Ray (GR)		

Values in parenthese are acceptable aliases. All matches are cases insensitive.

Range-List Parameters

Some parameters (for example BAND and TIME) may allow a parameter value to be specified as a numeric range. Such range-valued parameters use the forward slash (/) character as the separator between elements of the range specification (as in the ISO 8601 date specification after which this convention is patterned). For example, *5E-7/8E-7* would specify a range consisting of all values from 5E-7 to 8E-7, inclusive. If a third field is specified it is a step size for traversing the indicated range. If a parameter permits a step size the semantics of the step size are defined by the specific parameter.

An open range may be specified by omitting either range value. If the first value is omitted the range is open toward lower values. If the second value is omitted the range is open toward higher values. Omitting both values indicates an infinite range which accepts all values. For example, */5* is an open range which accepts all values less than or equal to 5. To specify all values less than 5, */4* would be used (for an integer valued range). Range values are limited to numeric values or ISO dates.

A list may be qualified by appending the character *;* (semicolon) followed by a qualifier string. For example *1E-7/3E-6;source* could specify a spectral bandpass in the rest frame of the source. List and range syntax may be combined, e.g., to indicate a list of scalar or range-valued parameter values. Such a range list may be ordered or unordered, and may contain either numeric or string data. An ordered list is one which requires values to be processed in a specified order, and to ensure this the range list is sorted or ordered by the service as necessary before being used. It is the responsibility of the service to sort an ordered range list, hence the client can input ranges or range values in any order for an ordered range list and the result will be the same. The sequence in which items in an unordered list occur on the other hand is significant, as since there is no intrinsic ordering for the list which can be enforced by the service, items will be processed by the service in the order they are input by the client.

TIME and BAND are typical examples of ordered range lists. Since a dataset matches the query if it contains data in any of the specified ranges, logically it does not matter in what order the ranges are given, or whether the first element of a range is less than the second, or whether ranges overlap; the result should be the same in all cases. Hence the range list has an intrinsic ordering irrespective of how ranges are input. Unless otherwise specified in the definition of a given parameter, range lists are assumed to be ordered.

VOCLIENT DAEMON PROCESSING

All VO-CLI tasks are built upon the VOClient interface an rely on a separate *vocliientd* process to provide the VO functionality. The vocliientd task is distributed as part of VO-CLI and will be started automatically by each task if it is not already running. If problems are encountered, you may want to manually start the vocliientd in a separate window before running the task so you can monitor the output for error messages.

RESOURCE CACHING

Directory resolution is a common activity of VO-CLI tasks and so results will be cached in the $HOME/.voclient/cache/regResolver directory based on the search term, service type and bandpass parameters. Defining the *VOC_NO_CACHE* environment variable will cause the task to ignore the cache.

EXAMPLES

1) Query the GSC 2.3 catalog for stars a) within the 0.1 degree default search size around NGC 1234: b) around all positions contained in file 'pos.txt': c) for the list of objects given on the command line: d) query a list of services for a list of positions: e) print a count of results that would be returned from 3 services for each position in a file:

```
% vodata gsc2.3 ngc1234                 (a)
% vodata gsc2.3 pos.txt                 (b)
% vodata gsc2.3 m31,m51,m93             (c)
% vodata svcs.txt pos.txt               (d)
% vodata hst,chandra,gsc2.3 pos.txt     (e)
```

2) Query all (142) image services having data of the subdwarf galaxy IC 10, print a count of the results only:

```
% vodata -c -t image any IC10
% vodata --count --type=image any IC10
```

Note that we use the reserved word '*any*' for the service name and constrain by image type. The task will automatically query the Directory to create the list of services to be queried.

3) Print a count of X-ray catalog data around Abell2712:

```
% vodata -c -t catalog -b x-ray any abell2712
% vodata --count --type=catalog --bandpass=x-ray any abell2712
```

In this case we constrain both the service type as well as the spectral coverage published for the resource in the Directory. We use the reserved flag to print a count without saving results. The object name is resolved to coordinates internally. (Note: this example may take a while to run).

4) Print the column metadata returned by the RC3 catalog service:

```
% vodata --meta rc3   or   vodata -m rc3
```

The output will print the result using the default VERBOSE level, adding the *-v* will set the query parameter VERBOSE=2, adding *-vv* will set VERBOSE=3 (to print all available columns). When accessing data the same *-v* flags will be required to retrieve columns at that *VERBOSE* level.

5) Use the Directory to query for resources using the search terms "cooling flow". Upon examining the output the user notices a Vizier paper titled "*Cooling Flows in 207 clusters of Galaxies*" that looks interesting. Use the **vodata** task to download all tables associated with this paper, save tables in the default CSV format:

```
% vodirectory cooling flow
% vodata -O white97 -all J/MNRAS/292/419/
% vodata --output=white97 --all J/MNRAS/292/419/
```

All 7 tables will be written to the current directory to files having a root name 'white97' (chosen based on the author and publication date).

6) Find a suitable XMM image service, get a (brief) count of the XMM images available for 3c273, and if there aren't too many, download the images and save the extracted access URLs:

```
% vodirectory -rv -t image xmm
ShortName      ResourceType    Title
-----------------------------------------------------....
XMM-Newton     SIAP/ARCHIVE    XMM-Newton Archive ....

% vodata -cq xmm-newton 3c273
  xmm-newton        27    I  XMM-Newton Archive ....

% vodata --count --quiet xmm-newton 3c273
  xmm-newton        27    I  XMM-Newton Archive ....

% vodata --get xmm-newton 3c273
   .... will query and download 27 images.
```

7) Query for the images available from 2MASS at a given position, extract the positions and service URLs to separate files:

```
% vodata -e -O 2mass -t image 2mass 12:34:56.7 -23:12:45.2
% vodata -e --output=2mass --type=image 2mass 12:34:56.7 -23:12:45.2
```

The query produces files with the root name '2mass', and exten- sions of ".*csv*" (the main response), ".*pos*" (the extracted pos- itions), and ".*urls*" (the access references). The user inspects the files and notices that the references return both FITS and HTML files, but she only wants the FITS image date and uses **vodata** to download only those:

```
% grep fits 2mass_I_001_15998.urls > images.txt
% vodata images.txt
```
or
```
% grep fits 2mass_I_001_15998.urls | vodata -i -
```

In both cases we pass URLs to the task which bypasses the query and directly access the images. However, in the first case we process the entire list and are able to take advantage of the *-maxdownloads=<N>* option to increase the number of simultaneous downloads. In the second case, the *-i* flag causes the task to interpret each line of the input stream as a separate command and so the data are always downloaded one at a time (which is the default download behavior anyway).

8) Use **vodata** as a test client for a locally-installed SIAP service:

```
% vodata -t image -s http://localhost/siap.pl 180.0 0.0
% vodata --type=image --svc=http://localhost/siap.pl 180.0 0.0
```

In this case we force the ServiceURL using the '*-s*' flag, but since we can't do a Directory query to discover what type of service this is, we must use the '*-t*' flag to indicate it is an image service.

9) Create a local table containing the Abell catalog. Begin with a Directory query to find likely services using the **vodirectory** task, print a verbose description of each resource and page the results with *less*:

```
% vodirectory -v -v --type=catalog abell | less
```

The verbose results indicate a number of services with differing requirements for what is included. We decide to use the service from HEASARC since it contains southern hemisphere data and constraints we are interested in. Try an all-sky query to retrieve the entire catalog, use the service identifier to be specific about where we want to go:

```
% vodata -e ivo://nasa.heasarc/abell 0.0 0.0 180.0
% vodata --extract ivo://nasa.heasarc/abell 0.0 0.0 180.0
```

We use the '*-e*' flag to extract the positions to a separate file with a ".*pos*" extension so that we can use these in later queries. However, the *.pos* file additionally contains the ID from the original catalog in column 1. Strip this column so we're left with only RA and DEC and query for Chandra observations at each position:

```
% cut -c6- *.pos | vodata ivo://nasa.heasarc/chanmaster -p -
% cut -c6- *.pos | vodata ivo://nasa.heasarc/chanmaster --pos=-
```

Here we used the unix '*cut*' utility to remove the first column and pipe the resulting positions to the task, using the '*-p -*' option to indicate positions should be read from stding, and the IVO identifier of the Chandra observation master log we also discovered from the Directory.

10) Interactively query for a count of Chandra observations of Messier objects:

```
% vodata -cq chandra -i -
m31
 chandra       335    I  Chandra X-Ray Observatory Data Archive
   :            :        :           :      :
```

Note that by using the '*-i*' flag we execute each query as if we'd put the object name on the command line. Using the '*-o*' flag would instead read all of the objects from the stdin but then execute the queries in parallel following an EOF to terminate the input.

11) Use the *STILTS* task '*tpipe*' to select rows from a VOTable of QSOs (made with an earlier query) where the redshift is > 0.2. Output only the positions and pipe this to **vodata** to generate a new query to see whether HST has observed any of these objects:

```
% stilts tpipe ifmt=votable qso_survey.vot
    cmd='select "Z > 0.2"'
  cmd='keepcols "RA DEC"' | vodata -p - hstpaec
```

Note that we use the '*-p -*' flag to tell the task to take it's list of positions from the piped input. The positions are used to query the HST Planned and Archived Exposure Catalog (*hstpaec*)

12) The user has a task called '*wcsinfo*' that takes a list of images and outputs a text table containing the plate center and size in degrees. Use this task as part of a query for 2MASS point sources contained in each image:

```
% wcsinfo *.fits | vodata 2mass-psc -i -
```

Here we specify the desired service (*2mass-psc*) on the commandline as usual, and allow the remainder of the args (i.e. the position and search size) to be read from the stdin. This allows for variable search sizes but processes the positions serially. If the sizes are all the same (say 25 arcmin), multiple queries can be done simultaneously using just a position file, e.g.

```
% wcsinfo -pos_only *.fits > centers.txt
% vodata --sr=25m 2mass-psc centers.txt
```

13) Query a large list of objects against a number of ASCA resources. Because the ASCA catalogs are served by the same machine, use the *'-i'* option and a command file to process only each resource sequentially while still parallelizing the object lists:

```
% cat cmds.txt
ASCA  survey.tbl
ASCA\ GIS  survey.tbl
ASCA\ GPS  survey.tbl
   :       :
```

```
% vodata -i cmds.txt
```

Note that we've needed to escape the space in the resource name in some cases. To avoid this, use of the IVO identifier for each resource is usually preferred.

14) Query the VO for GALEX data of M51. Because the *ShortName* GALEX is not unique, we must either specify the IVO identifier of a specific service to query, or if we're interested in results from all supported data services with *galex* in the name:

```
% vodata -a galex M51
% vodata --all galex M51
```

The results come from the Cone and SIAP services both called *GALEX*, as well as an additional SIAP service called 'GALEX_Atlas'. Note that the service names are case insensitive in either case.

15) Process a list of hundreds of positions against the GSC2.3 catalog:

```
% vodata gsc2.3 positions.txt
```

16) Process a list of hundreds of positions against the GSC2.3 catalog, assume that the input table has a 5 line header of text and three columns (id,ra,dec). Note that the *cols* option requires the optional id be specified last:

```
% vodata --cols=2,3,1 --hskip=5 gsc2.3 positions.txt
```

BUGS

Some services don't repond properly to the metadata query and will print a "no attributes found" message.

TODO

- Additional command-line options should be allowed to be specified in a command file.
- Support for SSAP, SAMP, TAP

Revision History

June 2007 - This task is new.

Author

Michael Fitzpatrick (fitz@noao.edu), June 2007

SEE ALSO

voclient(1), voclientd(1), vosesame(1), vodirectory(1)