Accessing Data from User Programs

Opening the Data Object

The call to open a data object is:

```
int maxlevel = jdbopen_(handle,obj,names,namesize,num)
   int *handle;
   char *obj;
   char *names;
   int *namesize;
   int *num;
```

Input Arguments:

`obj` The name of the data object. This looks like
`//machinename/directorypath/objectname(parameters)`
in the most general case.

`names` A list of names of variables. The string is supposed tohave space for `|num|*namesize` characters. The i^{th} name (beginning at 0) starts at offset i`*namesize` in the `names` area. If `num` is positive, then the names must be supplied and the values returned will be in the requested order. If `num` is negative, then the names array will be filled in and `num` will return the number of names present.

`num` The number of names supplied or the negative of the maximum number of names which the program has space for.

Returned values:

`handle` The open call returns an integer `handle` which is used in all subsequent calls to refer to this particular object.

`names` The variable names from `1..num`. Offsets as above.

`num` The number of variables.

`maxlevel` The maximum hierarchical depth of the data structure. If `maxlevel = -999`, there was an error in opening. If `maxlevel<0` then `names(|maxlevel|)` (which was requested) turned out not to be present.

Note: The indexing of names appears a bit confusing. Usually, however, one would simply set

```
char names[10][40];
int num= -10;
int namesize=40;
maxlevel=jdbopen_(obj,names,&namesize,&num);
```

and then just look at `names[i]`. So it's pretty straightforward.

Retrieving Comments

The call to get a string of comments from the data object is

```
int ok = jdbcomments_(handle,outcom)
   int *unit;
   char *outcom;
```

Input arguments

`handle` The handle for the desired object, as returned by `jdbopen`.

Returned values

`ok` The returned value is 1 (one) if comments are being returned and 0 (zero) if there are no more comment strings.

`comments` The comment string (if there is one). The string will be a maximum of 80 characters (including the zero terminator).

Generally, this call will be made just after the data object is opened (assuming you want the comments) and repeated until there are no more comments.

Variable Level

Data variables are arranged in a hierarchy, with the lest rapidly changing ones at the lowest (0) level and the most rapidly changing ones at the highest level (could be up to 9). Thus you might have `cruise` as a level 0 variable, `station, lat, lon, date` as level 1, and `press, temp, sal, o2` as level 2. To retrieve level information, use

```
int level = jdblevel_(handle,varnum)
   int *habdle;
   int *varnum;
```

Input arguments

`handle` The handle for the object.

`varnum` The number of the variable desired. This corresponds to the index into the `names` array (0..`num`−1).

Returned values

`level` The level for the variable requested.

Variable Attributes

The variable attributes are strings with descriptive material appropriate to the particular variable. Typically, units are encoded as an attribute. Additional attributes would be represented as *atributename=attributevalue*; e.g. `analysis_method=Winkler`.

```
int ok = jdbattributes_(handle,varnum,attr)
   int *handle;
   int *varnum;
   char *attr;
```

with the arguments

Input arguments

`handle` The handle for the object.

`varnum` The number of the variable desired. This corresponds to the index into the `names` array (0..`num`−1).

Returned values

`ok` The returned value is 1 (one) if attributes are being returned and 0 (zero) if there are no more attribute strings for this variable.

`attr` The attribute string (if there is one). The string will be a maximum of 80 characters (including the zero terminator).

Reading Real Numbers

To read the list of values corresponding to the named variables, you call

```
int level = jdbread_(handle,values)
   int *handle;
   float values[];
```

Input arguments

`handle` The handle for the object.

Returned values

`values` A vector of values corresponding to the list of names. The indices run from [0..`num`−1]. The special value -9999 is used for missing data.

`level` The minimum hierarchical depth which changed during this record. Thus if `level == maxlevel` then only the inner–most level changed. If `level` < `maxlevel` then a level further up has altered. This is useful information for breaking lines on plots, etc. A return value `level` < `0` indicates end–of–file on the object.

This is called repeatedly to get the values for each new record, until a return less than zero occurs. for each new record.

Reading ASCII Values

There is a similar call

```
int level=jdbreada_(handle,avalues,valuesize)
   int *handle;
   char *avalues;
   int *valuesize;
```

Input arguments

`handle` The handle for the object.

`valuesize` The length in characters reserved for each value string. (See `namesize` in `jdbopen`.)

Returned values

`avalues` A vector of values corresponding to the list of names. The offsets run from [0..`num`−1]*`valuesize`. The special value `nd` is used for missing data. See the discussion of `names` in the `jdbopen` call for further discussion of the way strings are returned.

`level` The minimum hierarchical depth which changed during this record. Thus if `level == maxlevel` then only the inner–most level changed. If `level` < `maxlevel` then a level further up has altered. This is useful information for breaking lines on plots, etc. A return value `level` < `0` indicates end–of–file on the object.

Closing the Object

To close, call

```
jdbclose_(handle)
   int *handle;
```

with the argument

Input argument

`handle` The handle for the object.

This can be done before the end–of–file on the data object if desired.

f77 Calls

```
maxlevel = jdbopen(ihandle,obj,names,namesize,num)
   integer jdbopen
   integer ihandle
   character*256 obj
   character*namesize names(|num|)
   integer namesize
   integer num
   integer maxlevel

iok = jdbcomments(ihandle,outcom)
   integer jdbcomments
   integer ihandle
   character*80 outcom

ilevel = jdblevel(ihandle,ivarnum)
   integer jdblevel
   integer ihandle
   integer ivarnum

iok = jdbattributes(ihandle,ivarnum,attr)
   integer jdbattributes
   integer ihandle
   integer ivarnum
```

```
   character*80 attr

ilevel = jdbread(ihandle,values)
   integer jdbread
   integer ihandle
   real values(num)

ilevel = jdbreada(handle,avalues,ivalsiz)
   integer jdbreada
   integer ihandle
   character*ivalsiz avalues(num)
   integer ivalsiz

call jdbclose(ihandle)
   integer ihandle
```