DRAFT of Proposed Data Standard for the APS

Jon Tischler, ORNL & UNI-CAT

INTRODUCTION

For the past ~15 years I have been taking data at synchrotron radiation facilities. In that period I have dealt with approximately 10 different data file "standards". Each standard was, of course, unique to a particular program. At first, writing a program to read in the data file and output one that I could then plot was interesting; but about 12 years ago I stopped enjoying it. 10 years ago I presented a short paper at the 3rd US Synch. Rad. Instr. Conf. (at BNL) describing a possible data standard based on the ISO-8211 (Specification of a data descriptive file for information interchange). This was thoroughly ignored; even I did not implement it. Although the ISO-8211 standard is still there, the HDF data standard now exists, and it is supported by the National Center for Supercomputing Applications (NCSA), and many data analysis programs. The existence of HDF libraries in C and FORTRAN for many different computer systems (UNIX's, Mac, DOS,...) make it an obvious first choice. With existing libraries, we have less to do, and it makes implementing the standard both easier and more rigorous.

Other standards exist for data files. The netCDF standard provides elements that make it very attractive for storing experimental data. This includes the ability to assign units to data, and other informative attributes. Unfortunately netCDF provides no way to group data into scans. There is no mechanism for measuring 6 scans of {theta, 2theta, scintillator, and clock}. When you name a variable "theta" in a netCDF file, that name is global to the entire file.

The HDF standard stores values as either arrays (of any dimension called an SDS) or tables (called a vdata). Since most of the scanned data appears to be tables (t ble of "theta", "scint"), this at first appears to be a good choice. However, vdatas have no means for identifying things such as the rank of the data, axes, or units, etc. HDF, also has something called a vgroup, which is way to group SDS's (and anything else) together. This allows HDF to form data into scans, so multiple scans can be put into one file.

This leaves us with a quandary, netCDF is good at describing data and HDF is good at organizing data, but not vice versa. That was the situation until early this year. The new version of HDF (HDF3.3r3) allows an SDS to look just like a netCDF variable with the ability to set dimension scales, attributes (like units) etc. Plus you can still group these SDS's into scans (which will be called "entrys") to put multiple scans into one file.

HDF DOCUMENTATION

Documentation for HDF is available via anonymous ftp from `ftp.ncsa.uiuc.edu`, in:

```
/Documentation/HDF.Vset2.1/
/Documentation/HDF_getting_started/
/Documentation/HDF3.3/
and the code is in /HDF/HDF3.3r3/
```

It is important to read the "HDF_getting_started" and the "HDF.Vset" manuals. Without Vset, HDF is rather useless, since it can only store images, large groups of numbers, and text. It needs Vset to specify relations between things (like grouping pieces of information together into a scan).

In writing this standard, the deciding factor at all times has been to organize the data so that it can be easily read and plotted by a generic program (not just by a specialized program customized to a particular experiment). This was the key factor used when closing among the various approaches possible with HDF.

CREATION of the DATA FILE

The method chosen for data storage is designed to make as much use as possible of the HDF standard which reduces the number of extra definitions. Thus just about everything is to be stored as a Scientific Data Set (SDS). And the individual scans are to be Vgroups with the sequential names entry1, entry2, entry3,...

The steps for making a data file are:

I) Open and initialize the file for writing for both Vgroups and SDS.
   A) Create Vgroup named "entry_default"
      1) Insert SDS's with global defaults
      2) detach "entry_default"
   B) Create a Vgroup named "entry1"
      1) Add SDS's for simple parameters (not scanned) and add to "entry1"
      2) Create Vgroup "data1" to hold the scanned information
         a) Create an SDS for each scanned variable (add appropriate attributes to it)
         b) add special attributes (axis, signal, ...) to scanned variables
         c) detach "data1"
      3) detach "entry1"
   C) Create a Vgroup named "entry2"
      1) repeat steps 1-3 from B
   D) Continue creating and filling "entry*n*" until done
   E) Close the file

OUTLINE of a TYPICAL DATA FILE

Useful variables to predefine, since they are used frequently.

```
#define VERSION  "0.1 alpha"
#define LOCATION "NSLS:X15A"
#define PROG_NAM "ORDIF"
#define MAX_SDS_LEN 23          /* maximum allowed length of an SDS name */
#define MAX_MOTOR_LEN 27        /* maximum allowed length of a motor name*/
int32 dim1[1]={1};
int32 edge1[1]={1};
int32 start0={0,0,0,0,0,0,0,0,0};/* max possible rank defined at start*/
int32 all_ones = -1;      /* used to flag non-existent data */
int16 one16 = 1;      /* a 16 bit one, used with DFNT_INT16 */
char  *file_name = "pdq_may20.hdf";
char  *user_name = "John Q. Public";
char  *user_mail = "Chemistry Dept.\nBerkeley U.\n CA  11111";
char  *user_phone = "900-555-1212";
char  *user_fax = "900-555-1213";
```

The first step in creating a data file is to open it, and set it up for HDF access by both the SD and Vgroup interfaces.

```
sfid = SDstart(file_name, DFACC_CREATE);  /*file name is "pdq_may20.hdf" */
vfid = Hopen(file_name,DFACC_RDWR,0);
Vstart(vfid);
```

Add global attributes to the file that apply to the *entire* file.

```
i = SDsetattr(sfid,"file_name",DFNT_CHAR8,strlen(file_name),file_name);
i = SDsetattr(sfid,"version",DFNT_CHAR8,strlen(VERSION),VERSION);
i = SDsetattr(sfid,"MAX_SDS_LEN",DFNT_INT16,1,MAX_SDS_LEN);
```

```
i = SDsetattr(sfid,"MAX_MOTOR_LEN",DFNT_INT16,1,MAX_MOTOR_LEN);
```

Create the Vgroup named "entry_default" to hold all of the information for the first entry.

```
vgN = Vattach(vfid,-1,"w");
Vsetname(vgN,"entry_default");   /* this name is required */
Vsetclass(vgN,"APS_default");    /* this class is required */
id = -1;
```

Create the SDS's to get default values.

```
sid[++id] = sdstring(sfid,"location",LOCATION); /* required */
sid[++id] = sdstring(sfid,"user_name",user_name);  /* required */
i = SDsetattr(sid[id],"user_mail",DFNT_CHAR8,strlen(user_mail),user_mail);
i = SDsetattr(sid[id],"user_email",DFNT_CHAR8,strlen(user_email),user_email);
i = SDsetattr(sid[id],"user_phone",DFNT_CHAR8,strlen(user_phone),user_phone);
i = SDsetattr(sid[id],"user_fax",DFNT_CHAR8,strlen(user_fax),user_fax);
sid[++id] = sdstring(sfid,"program_name",PROG_NAM);
sid[++id] = sdstring(sfid,"diffractometer","eulerian");
```

Group the SDS's into the entry_default group.

```
for (i=0;i<=id;i++) Vaddtagref(vgN,DFTAG_SDG,SDidtoref(sid[i]));
Vdetach(vgN);
```

Create a Vgroup named "entry1" to hold all of the information for the first entry.

```
vgN = Vattach(vfid,-1,"w");
Vsetname(vgN,"entry1");
Vsetclass(vgN,"APS_entry");      /* this class is required */
id = -1;
```

Create an SDS to describe certain general properties of this particular scan:

```
sid[++id] = sdstring(sfid,"date","22-feb-1996");     /* required */
sid[++id] = sdstring(sfid,"hour","23:59:59.12 UT");      /* required */
sid[++id] = sdstring(sfid,"entry_analysis","diffraction/misc");
sid[++id] = sdstring(sfid,"entry_intent","data");   /* required */
sid[++id] = sdstring(sfid,"sample","BiAsO on graphite #12345");
sid[++id] = sdstring(sfid,"title","First data on compound");

hutch_pressure = 1.001;
sid[++id] = SDcreate(sfid,"hutch_pressure",DFNT_FLOAT32,1,dim1);
SDsetdatastrs(sid[id],"barometric pressure","atmosphere","%.3","");
SDwritedata(sid[id],start0,NULL,dim1,&hutch_pressure);

hutch_temperature = 27.5;
sid[++id] = SDcreate(sfid,"hutch_temperature ",DFNT_FLOAT32,1,dim1);
SDsetdatastrs(sid[id],"room temperature","Celsius","%.2","");
SDwritedata(sid[id],start0,NULL,dim1,&hutch_pressure);

temperature = 100.5;
sid[++id] = SDcreate(sfid,"temperature",DFNT_FLOAT32,1,dim1);
SDsetdatastrs(sid[id],"temperature","Kelvin","%.3","");
SDwritedata(sid[id],start0,NULL,dim1,&temperature);

mono_energy = 8.047;
sid[++id] = SDcreate(sfid,"mono_energy",DFNT_FLOAT32,1,dim1);
SDsetdatastrs(sid[id],"energy","keV","%.4","");
SDwritedata(sid[id],start0,NULL,dim1,&mono_energy);
```

Make a group to hold the scan, and write the scan data. This is a theta-2theta scan with 50 intervals(51 points). The scan range is from theta=[10,20], and 2theta=[20,40].

```
vgD = Vattach(vfid,-1,"w");
Vsetname(vgD,"data1");   /* ALWAYS "data1", even for "entry2" */
Vsetclass(vgD,"APS_scan");   /* this class name required */
Did = -1;
dims[0] edge[0] = = 51;  /* scan of 51 points, rank is 1 */

Dsid[++Did] = SDcreate(sfid,"theta",DFNT_FLOAT32,1,dims);
SDsetdatastrs(Dsid[Did],"theta","degrees","%.3","");
lo = 10.; hi = 20.;  SDsetrange(Dsid[Did],&lo.,&hi);
SDsetattr(Dsid[Did],"axis",DFNT_INT16,1,&one16);
SDsetattr(Dsid[Did],"diffract_axis",DFNT_CHAR8,1,"theta");
SDwritedata(Dsid[Did],start0,NULL,edge,column1);

Dsid[++Did] = SDcreate(sfid,"2theta",DFNT_FLOAT32,1,dims);
SDsetdatastrs(Dsid[Did],"2theta","degrees","%.3","");
lo = 20.; hi = 40.;  SDsetrange(Dsid[Did],&lo,&hi);
SDsetattr(Dsid[Did],"diffract_axis",DFNT_CHAR8,1,"2theta");
SDwritedata(Dsid[Did],start0,NULL,edge,column2);

gain = 1.e8;
Dsid[++Did] = SDcreate(sfid,"ic1",DFNT_INT32,1,dims);
SDsetattr(Dsid[Did],"gain",DFNT_FLOAT32,1,&gain);
SDsetattr(Dsid[Did],"I_monitor",DFNT_NONE,1,NULL);
SDsetdatastrs(Dsid[Did],"photons at ic1","photons","%.3","");
SDsetcal(Dsid[Did],12345.,0.,22.,0.,DFNT_FLOAT32);
/* calibration contains ic gain and absorption to give photons */
SDsetattr(Dsid[Did],"serial_no",DFNT_CHAR8,strlen("1234-AX"),"1234-AX");
SDsetfillvalue(Dsid[Did],&all_ones);
SDwritedata(Dsid[Did],start0,NULL,edge,column3);

absorption_corr = 0.237;  deadtime=1.e-6;
Dsid[++Did] = SDcreate(sfid,"scint",DFNT_INT32,1,dims);
SDsetattr(Dsid[Did],"primary",DFNT_INT16,1,&one16);
SDsetattr(Dsid[Did],"signal",DFNT_INT16,1,&one16);
SDsetattr(Dsid[Did],"absorption_corr",DFNT_FLOAT32,1,&absorption_corr);
SDsetattr(Dsid[Did],"deadtime",DFNT_FLOAT32,1,&deadtime);
SDsetdatastrs(Dsid[Did],"lambda","photons","%.0","");
SDsetfillvalue(Dsid[Did],&all_ones);
SDwritedata(Dsid[Did],start0,NULL,edge,column4);

Dsid[++Did] = SDcreate(sfid,"clock",DFNT_INT32,1,dims);
SDsetattr(Dsid[Did],"count_time",DFNT_NONE,1,NULL);
SDsetdatastrs(Dsid[Did],"clock","1/262144 seconds","%.3","");
SDsetfillvalue(Dsid[Did],&all_ones);
SDwritedata(Dsid[Did],start0,NULL,edge,column5);
   /* attach SD's together to make data1*/
for (i=0;i<=Did;i++) Vaddtagref(vgD,DFTAG_SDG,SDidtoref(Dsid[i]));
Vdetach(vgD);
   /* attach SD's and vgD to make entry1*/
for (i=0;i<=id;i++) Vaddtagref(vgN,DFTAG_SDG,SDidtoref(sid[i]));
Vaddtagref(vgN,DFTAG_VG,vgD);
Vdetach(vgN);
```

Repeat for each following entry

```
vgN = Vattach(vfid,-1,"w");
Vsetname(vgN,"entry2");
Vsetclass(vgN,"APS_entry");
id=0;

etc...

for (i=0;i<=id;i++) Vaddtagref(vgN,DFTAG_SDG,SDidtoref(sid[i]));
Vaddtagref(vgN,DFTAG_VG,vgD);
Vdetach(vgN);
```

Finally close the file

```
SDend(sfid);
Vend(vfid);
Hclose(vfid);
```

Column names are set by each station for what ever is convenient. However, for the automatic plotting and analysis of data, it is necessary to identify certain columns that have special meaning, such as the axes and the primary, signal, and the count time for a point. This is done with predefined attributes that are attached the appropriate SDS. In the example above the following five lines identify certain SDS's as the preferred x-axis, signal, etc.

```
SDsetattr(sid[id],"axis",DFNT_INT16,1,&one16);
SDsetattr(sid[id],"primary",DFNT_INT16,1,&one16);
SDsetattr(sid[id],"signal",DFNT_INT16,1,&one16);
SDsetattr(sid[id],"count_time",DFNT_NONE,1,NULL);
SDsetattr(sid[id],"I_monitor",DFNT_NONE,1,NULL);
```

The full list of predefined attributes are listed in Appendix C. Note how the counting time was named "clock" but identified by the attribute "count_time" and the time in seconds was set by a "units" attribute to "sec". Also, the detector column was identified as both "primary" and "signal". The attribute "primary" flags the preferred variable(s) for plotting and "signal" is the measured variable with the closest relation to "primary". These two functions are separate because you might prefer to plot a quantity derived from the signal. If primary had been chosen to be the signal normalized by the monitor and corrected for detector deadtime, the source code would be changed to.

```
Dsid[++Did] = SDcreate(sfid,"scint",DFNT_INT32,1,dims);
SDsetattr(Dsid[Did],"signal",DFNT_INT16,1,&one16);
SDsetdatastrs(Dsid[Did],"lambda","photons","%.0","");
SDsetfillvalue(Dsid[Did],&all_ones);
SDwritedata(Dsid[Did],start0,NULL,edge,column4);

for (i=0;i<dims[0];i++) {
   dt = 1. / (1. - exp(-column6[i]/262144./deadtime));
   column6[i] = column4[i]/column3[i] * gain * absorption_corr * dt;
}
Dsid[++Did] = SDcreate(sfid,"corrected_signal",DFNT_INT32,1,dims);
strcpy(equation,"lambda/ic1*gain*absorption_corr/(1.-exp(-column6[i]/262144./deadtime))");
SDsetattr(Dsid[Did],"equation",DFNT_CHAR8,strlen(equation),equation);
SDsetattr(Dsid[Did],"primary",DFNT_INT16,1,&one16);
SDsetdatastrs(Dsid[Did],"reflectivity","absolute","%.5","");
SDsetfillvalue(Dsid[Did],&all_ones);
SDwritedata(Dsid[Did],start0,NULL,edge,column6);
```