Subject: Re: IDL "expert" needs help BAD!
Posted by rivers on Thu, 23 Jan 1997 08:00:00 GMT
View Forum Message <> Reply to Message

Hi David,

> Several people have asked me about writing HDF files, so I
> thought I would sit down and write a short tutorial about
> it. Here is what I am trying to do:
...
> This would seem to be straightforward (even with the
> decidedly unhelpful IDL examples, which use an
> archaic API), but I am struggling. It has gotten to the
> point that I am swearing at my computer. :-)

I have not tried to do exactly what you are describing, but I am enclosing an example routine which I have sucessfully used to write an HDF file, and an IDL HDF browser which lets one browse the structure (but not much of the content) of HDF files which obey some conventions we are using. The program which writes the files is a pretty straight translation into IDL of a C program which was an example of how we propose to store data at the Advanced Photon Source.

Hope they are useful.

Mark Rivers (773) 702-2279 (office)
CARS (773) 702-9951 (secretary)
Univ. of Chicago (773) 702-5454 (FAX)
5640 S. Ellis Ave. (708) 922-0499 (home)
Chicago, IL 60637 rivers@cars.uchicago.edu (e-mail)

or:
Argonne National Laboratory (630) 252-0422 (office)
Building 434A (630) 252-0405 (lab)
9700 South Cass Avenue (630) 252-1713 (beamline)
Argonne, IL 60439 (630) 252-0443 (FAX)


```
;    HDF_EXAMPLES.PRO
function hiw, j
   return, j/65536L
end

function low, j
   return, j mod 65536L
end
```

```
function sdstring, sfid, sd_name, string
   ; this routine writes an SDS which only contains a single char string
   dim = [0L]
   dim(0) = strlen(string)
   sid = HDF_SD_CREATE(sfid, sd_name, /string, dim)
   ; NOTE - THERE IS A BUG IN IDL 4.0.1B - TEMPORARY() IN THE FOLLOWING CALL
   ; IS A WORKAROUND. IT SHOULD BE REMOVED WHEN THE BUG IS FIXED
   HDF_SD_ADDDATA, sid, temporary(string)
   return, sid
end


pro entry_log, entry, vgN, vgD, id, Did, sid, Dsid

if (entry ge 1) then str = "entry"+string(entry) else str = "entry_default"

print, "   The Vgroup ", str, " is id = ", hiw(vgN), low(vgN), $
       " and Vgroup 'data1' is id = ", hiw(vgD), low(vgD)
print, "   The SDS's in ", str, " are:"
for i=0, id-1 do begin
   ref = HDF_SD_IDTOREF(sid(i))
   print, sid(i), hiw(ref), low(ref)
endfor

if (Did lt 0) then print, "  There is no 'data1' for ", str $
   else print, "  The SDS's in 'data1' of ", str, " are:"
for i=0, Did-1 do begin
   ref =  HDF_SD_IDTOREF(Dsid(i))
   print, Dsid(i), hiw(ref), low(ref)
endfor
end


; This is the main program for HDF_EXAMPLES

BELL           = 7B       ; bell character
MAX_ID         = 50
DFTAG_NDG      = 720
DFTAG_VG       = 1965
LOCATION       = "APS:6-BM-R"
PROG_NAM       = "ORDIF"
VERSION        = '0.1 alpha'
MAX_SDS_LEN    = 23       ; maximum allowed length of an SDS name
MAX_MOTOR_LEN  = 27       ; maximum allowed length of a motor name

vgD            = 0L

all_ones       = -1                ; used to flag non-existant data
```

```
file_name       = "examples.hdf"
user_name       = "John Q. Public"
user_mail       = "Chemistry Dept.\nBerkeley U.\n CA  11111"
user_email      = "joe@pdp8.chem.mit.edu"
user_phone      = '900-555-1212'
user_fax        = '900-555-1213'
sid             = lonarr(MAX_ID)
Dsid            = lonarr(MAX_ID)

;  open the file for Vset and SD

print, "opening file"
sfid = HDF_SD_START(file_name, /CREATE)
vfid = HDF_OPEN(file_name,/RDWR)
;HDF_VG_START, vfid

; write global attributes to the file

print, "writing global attributes"
HDF_SD_ATTRSET, sfid, "file_name", file_name
HDF_SD_ATTRSET, sfid, "version", VERSION
HDF_SD_ATTRSET, sfid, "MAX_SDS_LEN", MAX_SDS_LEN
HDF_SD_ATTRSET, sfid, "MAX_MOTOR_LEN", MAX_MOTOR_LEN

; entry_default,  The default entry.
print
print, "starting 'entry_default'"
vgN = HDF_VG_ATTACH(vfid, -1, /WRITE)
; This class and name are required
HDF_VG_SETINFO, vgN, NAME="entry_default", CLASS="APS_default"
id = 0
sid(id) = sdstring(sfid,"location",LOCATION)
id = id + 1
sid(id) = sdstring(sfid,"program_name",PROG_NAM)
id = id + 1
sid(id) = sdstring(sfid,"user_name",user_name)
HDF_SD_ATTRSET, sid(id), "user_mail", user_mail
HDF_SD_ATTRSET, sid(id), "user_email", user_email
HDF_SD_ATTRSET, sid(id), "user_phone", user_phone
HDF_SD_ATTRSET, sid(id), "user_fax", user_fax
id = id + 1
sid(id) = HDF_SD_CREATE(sfid, "h", /FLOAT, [1])
;default, sets h1
HDF_SD_ATTRSET, sid(id), "diffract_axis", "h1"
id = id + 1
sid(id) = HDF_SD_CREATE(sfid, "k", /FLOAT, [1])
;default, sets k1
```

```
HDF_SD_ATTRSET, sid(id), "diffract_axis", "k1"
id = id + 1
sid(id) = HDF_SD_CREATE(sfid, "l", /FLOAT, [1])
;default, sets l1
HDF_SD_ATTRSET, sid(id), "diffract_axis", "l1"
id = id + 1
sid(id) = HDF_SD_CREATE(sfid, "theta", /FLOAT, [1])
;default, sets theta
HDF_SD_ATTRSET, sid(id), "diffract_axis", "theta"
id = id + 1
sid(id) = HDF_SD_CREATE(sfid, '2theta', /FLOAT, [1])
;default, set 2theta
HDF_SD_ATTRSET, sid(id), "diffract_axis", '2theta'
id = id + 1
sid(id) = HDF_SD_CREATE(sfid, "chi", /FLOAT, [1])
;default, sets chi
HDF_SD_ATTRSET, sid(id), "diffract_axis", "chi"
id = id + 1
sid(id) = HDF_SD_CREATE(sfid, "phi", /FLOAT, [1])
;default, sets phi
HDF_SD_ATTRSET, sid(id), "diffract_axis", "phi"
entry_log, -1, vgN, vgD, id, -1, sid, Dsid
for i=0, id do HDF_VG_ADDTR, vgN, DFTAG_NDG, HDF_SD_IDTOREF(sid(i))
for i=0, id do HDF_SD_ENDACCESS, sid(i)
HDF_VG_DETACH, vgN

; entry1,  The maximally minimal entry.
print
print, "starting 'entry1'"
vgN = HDF_VG_ATTACH(vfid, -1, /WRITE)
HDF_VG_SETINFO, vgN, name="entry1" , class="APS_entry"
id = 0
sid(id) = sdstring(sfid,"date",'22-feb-1996')
id = id + 1
sid(id) = sdstring(sfid,"hour",'23:59:59.00 UT')
id = id + 1
sid(id) = sdstring(sfid,"entry_analysis","NONE")
id = id + 1
sid(id) = sdstring(sfid,"entry_intent","misc")

entry_log, 1, vgN, vgD, id, -1, sid, Dsid
for i=0, id do HDF_VG_ADDTR, vgN, DFTAG_NDG, HDF_SD_IDTOREF(sid(i))
for i=0, id do HDF_SD_ENDACCESS, sid(i)
HDF_VG_DETACH, vgN

; entry2,  A simple EXAFS scan with log ratio stored in data file.
print
print, "starting 'entry2'"
```

```
; make a dummy scan of 501 points
mono_energy = 20. + .001 * findgen(501)
ic1 = lonarr(501) + 200000L
ic2 = 100000L + 2*lindgen(501)
dims = [501]                ; scan of 501 points, rank=1
gain1 = 1.e8
gain2 = 1.e8
vgN = HDF_VG_ATTACH(vfid, -1, /WRITE)
HDF_VG_SETINFO, vgN, name="entry2", class="APS_entry"
id = 0
sid(id) = sdstring(sfid,"date",'23-feb-1996')
id = id + 1
sid(id) = sdstring(sfid,"hour",'00:10:59.10 -7')
id = id + 1
sid(id) = sdstring(sfid,"entry_analysis","EXAFS")
id = id + 1
sid(id) = sdstring(sfid,"entry_intent","data")
id = id + 1
sid(id) = sdstring(sfid,"title","Mo K-edge")

vgD = HDF_VG_ATTACH(vfid, -1, /WRITE)
HDF_VG_SETINFO, vgD, name="data1", class="APS_scan"
Did = 0
Dsid(Did) = HDF_SD_CREATE(sfid, "mono_energy", /FLOAT, dims)
HDF_SD_SETINFO, Dsid(Did), coordsys="energy", unit="keV",format="%.5"
HDF_SD_ATTRSET, Dsid(Did), "axis", 1
HDF_SD_ADDDATA, Dsid(Did), mono_energy

Did = Did + 1
Dsid(Did) = HDF_SD_CREATE(sfid, "ic1", /LONG, dims)
HDF_SD_ATTRSET, Dsid(Did), "gain", gain1
HDF_SD_ATTRSET, Dsid(Did), "I_monitor", " "
HDF_SD_SETINFO, Dsid(Did), coordsys="monitor", unit="photons", format="%.5", $
        fill=all_ones
Did = Did + 1
Dsid(Did) = HDF_SD_CREATE(sfid, "ic2", /LONG, dims)
HDF_SD_ATTRSET, Dsid(Did), "gain", gain2
HDF_SD_ATTRSET, Dsid(Did), "signal", 1
HDF_SD_SETINFO, Dsid(Did), coordsys="detector", unit="photons", format="%.5, $
        fill = all_ones
HDF_SD_ADDDATA, Dsid(Did), ic2

computed = alog(float(ic2)/ic1)
Did = Did + 1
Dsid(Did) = HDF_SD_CREATE(sfid, "corrected_signal", /FLOAT, dims)
equation = "log(ic2/ic1)"
HDF_SD_ATTRSET, Dsid(Did), "equation", equation
HDF_SD_ATTRSET, Dsid(Did), "primary", 1
```

```
HDF_SD_SETINFO, Dsid(Did), coordsys="log(ic2/ic1)", format="%.5", $
            fill=all_ones
HDF_SD_ADDDATA, Dsid(Did), computed

entry_log, 2, vgN, vgD, id, Did, sid, Dsid
for i=0, Did do HDF_VG_ADDTR, vgD, DFTAG_NDG, HDF_SD_IDTOREF(Dsid(i))
for i=0, Did do HDF_SD_ENDACCESS, Dsid(i)
HDF_VG_DETACH, vgD
for i=0, id do HDF_VG_ADDTR, vgN, DFTAG_NDG, HDF_SD_IDTOREF(sid(i))
HDF_VG_ADDTR, vgN, DFTAG_VG, vgD
for i=0, id do HDF_SD_ENDACCESS, sid(i)
HDF_VG_DETACH, vgN

; entry3,  A theta scan of 10 intervals with one counter and one CCD image
; saved at each point (64 x 64).

print
PRINT, "starting 'entry3'"
; make a dummy scan of 11 points
theta = 10.+ findgen(11)
ic1 = lonarr(11) + 200000L
scint = 250L + lindgen(11)
ccd_data = intarr(11, 64, 64)
CCD_x_axis = findgen(64)*.5
CCD_y_axis = findgen(64)*.45
dims = [11,64,64]
for i=0, 10 do begin
   for ii=0, 63 do begin
      for iii=0, 63 do ccd_data(i, ii, iii) = 1000L*i + 100L*ii + iii
   endfor
endfor
vgN = HDF_VG_ATTACH(vfid, -1, /WRITE)
HDF_VG_SETINFO, vgN, name="entry3", class="APS_entry"
id = 0
sid(id) = sdstring(sfid,"date",'23-feb-1996')
id = id + 1
sid(id) = sdstring(sfid,"hour",'02:10:59.10 -7')
id = id + 1
sid(id) = sdstring(sfid,"entry_analysis","diffraction/misc")
id = id + 1
sid(id) = sdstring(sfid,"entry_intent","data")
id = id + 1
sid(id) = sdstring(sfid,"title","fancy CCD scan")

mono_energy = 8.047
id = id + 1
sid(id) = HDF_SD_CREATE(sfid, "mono_energy", /FLOAT, [1])
HDF_SD_SETINFO, sid(id), coordsys="energy", unit="keV", format="%.4"
```

```
HDF_SD_ADDDATA, sid(id), mono_energy

vgD = HDF_VG_ATTACH(vfid, -1, /WRITE)
HDF_VG_SETINFO, vgD, name="data1", class="APS_scan"
Did = 0

; the HDF_SD_ATTRSET,  for "diffract_axis" is redundant, see entry_default
Dsid(Did) = HDF_SD_CREATE(sfid, "theta", /FLOAT, [dims(0)])
HDF_SD_SETINFO, Dsid(Did), coordsys="theta", unit="degrees", format="%.3"
HDF_SD_ATTRSET, Dsid(Did), "axis", 1
HDF_SD_ATTRSET, Dsid(Did), "diffract_axis", "theta"
HDF_SD_ADDDATA, Dsid(Did), theta
Did = Did + 1
Dsid(Did) = HDF_SD_CREATE(sfid, "ic1", /LONG, [dims(0)])
HDF_SD_ATTRSET, Dsid(Did), "gain", gain1
HDF_SD_ATTRSET, Dsid(Did), "I_monitor", " "
HDF_SD_SETINFO, Dsid(Did), coordsys="monitor", format="%.5", fill=all_ones
HDF_SD_ADDDATA, Dsid(Did), ic1
Did = Did + 1
Dsid(Did) = HDF_SD_CREATE(sfid, "scint", /LONG, [dims(0)])
HDF_SD_ATTRSET, Dsid(Did), "I_monitor", " "
HDF_SD_SETINFO, Dsid(Did), coordsys="flourescence detector", unit="counts",$
            format="%.0", fill=all_ones
HDF_SD_ADDDATA, Dsid(Did), scint
Did = Did + 1
Dsid(Did) = HDF_SD_CREATE(sfid, "CCD", /INT, dims)
HDF_SD_ATTRSET, Dsid(Did), "signal", 1
HDF_SD_ATTRSET, Dsid(Did), "primary", 1
HDF_SD_SETINFO, Dsid(Did), coordsys="CCD detector", unit="photons", $
            format="%.5"
dim_id = HDF_SD_DIMGETID(Dsid(Did), 1)
HDF_SD_DIMSET, dim_id, scale=CCD_x_axis
dim_id = HDF_SD_DIMGETID(Dsid(Did), 2)
HDF_SD_DIMSET, dim_id, scale=CCD_y_axis
HDF_SD_SETINFO, Dsid(Did), fill=all_ones
HDF_SD_ADDDATA, Dsid(Did), ccd_data

; Note, for a CCD there is more information to stored.  So, it is best to
; create another Vgroup called "CCD_spec", and attach it to the Vgroup vgN
; (not to vgD).  In this new Vgroup make numerous simple SDS's, one for each
; piece of information.  The reason for using SDS's is so that you can attach
; dimensions to the information.  For example to store an sid which refers
; to a dark current exposure, or distance from the CCD to the sample, etc.

entry_log, 3, vgN, vgD, id, Did, sid, Dsid
for i=0, Did do HDF_VG_ADDTR, vgD, DFTAG_NDG, HDF_SD_IDTOREF(Dsid(i))
for i=0, Did do HDF_SD_ENDACCESS, Dsid(i)
HDF_VG_DETACH, vgD
```

```
for i=0, id do HDF_VG_ADDTR, vgN, DFTAG_NDG, HDF_SD_IDTOREF(sid(i))
HDF_VG_ADDTR, vgN, DFTAG_VG, vgD
for i=0, id do HDF_SD_ENDACCESS, sid(i)
HDF_VG_DETACH, vgN

; entry4,  A line in hkl space.  The scan direction is along the (111)

print
print, "starting 'entry4'"
; make a dummy scan of 51 points
theta = 10. + .02*findgen(51)
ttheta = 2*theta
chi = fltarr(51) + 45.
phi = fltarr(51) + 20.
h1 = 1. + findgen(51)*0.002
k1 = h1
l1 = h1
ic1  = lonarr(51)+ 2000000L
x = theta - 10.5
scint = long(3000. * exp(-(x*x/.1)))
dims = [51]
vgN = HDF_VG_ATTACH(vfid, -1, /WRITE)
HDF_VG_SETINFO, vgN, name="entry4", class="APS_entry"
id = 0
sid(id) = sdstring(sfid,"date",'25-Feb-1996')
id = id + 1
sid(id) = sdstring(sfid,"hour",'02:10:59.10 -7')
id = id + 1
sid(id) = sdstring(sfid,"entry_analysis","diffraction/misc")
id = id + 1
sid(id) = sdstring(sfid,"entry_intent","data")
id = id + 1
sid(id) = sdstring(sfid,"title","along the (111) direction")
id = id + 1
sid(id) = sdstring(sfid,"constraint","delta_h = delta_k = delta_l")

mono_energy = 8.047
id = id + 1
sid(id) = HDF_SD_CREATE(sfid, "mono_energy", /FLOAT, [1])
HDF_SD_SETINFO, sid(id), coordsys="energy", unit="keV", format="%.4"
HDF_SD_ADDDATA, sid(id), mono_energy

vgD = HDF_VG_ATTACH(vfid, -1, /WRITE)
HDF_VG_SETINFO, vgD, name="data1", class="APS_scan"
Did = 0

; the next seven calls to HDF_SD_ATTRSET,  for "diffract_axis" are redundant,
; see entry_default above
```

```
Dsid(Did) = HDF_SD_CREATE(sfid, "theta", /FLOAT, dims)
HDF_SD_SETINFO, Dsid(Did), coordsys="theta", unit="degrees", format="%.3"
HDF_SD_ATTRSET, Dsid(Did), "diffract_axis", "theta"
HDF_SD_ADDDATA, Dsid(Did), theta
Did = Did + 1
Dsid(Did) = HDF_SD_CREATE(sfid, '2theta', /FLOAT, dims)
HDF_SD_SETINFO, Dsid(Did), coordsys='2theta', unit="degrees", format="%.3"
HDF_SD_ATTRSET, Dsid(Did), "diffract_axis", '2theta'
HDF_SD_ADDDATA, Dsid(Did), ttheta
Did = Did + 1
Dsid(Did) = HDF_SD_CREATE(sfid, "chi", /FLOAT, dims)
HDF_SD_SETINFO, Dsid(Did), coordsys="chi", unit="degrees", format="%.3"
HDF_SD_ATTRSET, Dsid(Did), "diffract_axis", "chi"
HDF_SD_ADDDATA, Dsid(Did), chi
Did = Did + 1
Dsid(Did) = HDF_SD_CREATE(sfid, "phi", /FLOAT, dims)
HDF_SD_SETINFO, Dsid(Did), coordsys="phi", unit="degrees", format="%.3"
HDF_SD_ATTRSET, Dsid(Did), "diffract_axis", "phi"
HDF_SD_ADDDATA, Dsid(Did), phi
Did = Did + 1
Dsid(Did) = HDF_SD_CREATE(sfid, "h1", /FLOAT, dims)
HDF_SD_SETINFO, Dsid(Did), coordsys="h", format="%.3"
HDF_SD_ATTRSET, Dsid(Did), "diffract_axis", "h1"
HDF_SD_ADDDATA, Dsid(Did), h1
Did = Did + 1
Dsid(Did) = HDF_SD_CREATE(sfid,"k1",/FLOAT, dims)
HDF_SD_SETINFO, Dsid(Did), coordsys="k", format="%.3"
HDF_SD_ATTRSET, Dsid(Did), "diffract_axis", "k1"
HDF_SD_ADDDATA, Dsid(Did), k1
Did = Did + 1
Dsid(Did) = HDF_SD_CREATE(sfid, "l1", /FLOAT, dims)
HDF_SD_SETINFO, Dsid(Did), coordsys="l", format="%.3"
HDF_SD_ATTRSET, Dsid(Did), "diffract_axis", "l1"
HDF_SD_ADDDATA, Dsid(Did), l1

computed = sqrt(h1^2 + k1^2 + l1^2)
Did = Did + 1
Dsid(Did) = HDF_SD_CREATE(sfid, "x-axis", /FLOAT, dims)
equation = "sqrt(h*h+k*k+l*l)"
HDF_SD_ATTRSET, Dsid(Did), "equation", equation
HDF_SD_ATTRSET, Dsid(Did), "axis", 1
HDF_SD_SETINFO, Dsid(Did), coordsys="distance along (111)", format="%.5", $
        fill = all_ones
HDF_SD_ADDDATA, Dsid(Did), computed
Did = Did + 1
Dsid(Did) = HDF_SD_CREATE(sfid, "ic1", /LONG, dims)
HDF_SD_ATTRSET, Dsid(Did), "gain", gain1
HDF_SD_ATTRSET, Dsid(Did), "I_monitor", " "
```

```
HDF_SD_SETINFO, Dsid(Did), coordsys="monitor", unit="photons", format="%.5", $
          fill = all_ones
HDF_SD_ADDDATA, Dsid(Did), ic1
Did = Did + 1
Dsid(Did) = HDF_SD_CREATE(sfid, "scint", /LONG, dims)
HDF_SD_ATTRSET, Dsid(Did), "signal", 1
HDF_SD_ATTRSET, Dsid(Did), "primary", 1
HDF_SD_SETINFO, Dsid(Did), coordsys="detector", unit="photons", format="%.5", $
          fill = all_ones
HDF_SD_ATTRSET, Dsid(Did), "analyzer", "Ge111"
HDF_SD_ADDDATA, Dsid(Did), scint

entry_log, 4, vgN, vgD, id, Did, sid, Dsid
for i=0, Did do HDF_VG_ADDTR, vgD, DFTAG_NDG, HDF_SD_IDTOREF(Dsid(i))
for i=0, Did do HDF_SD_ENDACCESS, Dsid(i)
HDF_VG_DETACH, vgD
for i=0, id do HDF_VG_ADDTR, vgN, DFTAG_NDG, HDF_SD_IDTOREF(sid(i))
HDF_VG_ADDTR, vgN, DFTAG_VG, vgD
for i=0, id do HDF_SD_ENDACCESS, sid(i)
HDF_VG_DETACH, vgN

; entry5,  A surface in hkl space.  The scan directions are along
; the (100) and (011)
print
print, "starting 'entry5'"
h1a = fltarr(21, 21)
k1a = fltarr(21, 21)
l1a = fltarr(21, 21)
ic1a = lonarr(21, 21)
scint = lonarr(21, 21)
for j=0, 20 do begin        ; make a dummy scan of 21 x 21 points
   for i=0, 20 do begin
      h1a(i,j) = 1. + .1*j
      k1a(i,j) = 1. + .05*i
      l1a(i,j) = k1a(i,j)
      ic1a(i,j) = 20000
      x = (i-10)*(i-10) /3. + (j-10)*(j-10)/5.
      scint(i) = 1000. * exp(-x)
   endfor
endfor
dims=[21,21]
vgN = HDF_VG_ATTACH(vfid, -1, /WRITE)
HDF_VG_SETINFO, vgN, name="entry5", class="APS_entry"
id = 0
sid(id) = sdstring(sfid,"date",'25-Feb-1996')
id = id + 1
sid(id) = sdstring(sfid,"hour",'03:10:59.10 -7')
id = id + 1
```

```
sid(id) = sdstring(sfid,"entry_analysis","diffraction/misc")
id = id + 1
sid(id) = sdstring(sfid,"entry_intent","data")
id = id + 1
sid(id) = sdstring(sfid,"title","the perpendicular surface")
id = id + 1
sid(id) = sdstring(sfid,"constraint","perpendicular to (0 -1 1)")

mono_energy = 8.047
id = id + 1
sid(id) = HDF_SD_CREATE(sfid,"mono_energy",/FLOAT,[1])
HDF_SD_SETINFO, sid(id), coordsys="energy", unit="keV", format="%.4"
HDF_SD_ADDDATA, sid(id), mono_energy
vgD = HDF_VG_ATTACH(vfid, -1, /WRITE)
HDF_VG_SETINFO, vgD, name="data1", class="APS_scan"
Did = 0

Dsid(Did) = HDF_SD_CREATE(sfid, "h1", /FLOAT, dims)
HDF_SD_SETINFO, Dsid(Did), coordsys="h", format="%.3"
; diffract_axis attribute set in entry_default
HDF_SD_ATTRSET, Dsid(Did), "axis", 1
HDF_SD_ADDDATA, Dsid(Did), h1a
Did = Did + 1
Dsid(Did) = HDF_SD_CREATE(sfid, "k1", /FLOAT, dims)
HDF_SD_SETINFO, Dsid(Did), coordsys="k", format="%.3"
; diffract_axis attribute set in entry_default
HDF_SD_ADDDATA, Dsid(Did), k1a
Did = Did + 1
Dsid(Did) = HDF_SD_CREATE(sfid, "l1", /FLOAT, dims)
HDF_SD_SETINFO, Dsid(Did), coordsys="l", format="%.3"
; diffract_axis attribute set in entry_default
HDF_SD_ADDDATA, Dsid(Did), l1a
computeda = sqrt(k1a^2 + l1a^2)
Did = Did + 1
Dsid(Did) = HDF_SD_CREATE(sfid, "y-axis", /FLOAT, dims)
equation="sqrt(k*k+l*l)"
HDF_SD_ATTRSET, Dsid(Did), "equation", equation
HDF_SD_ATTRSET, Dsid(Did), "axis", 2
HDF_SD_SETINFO, Dsid(Did), coordsys="distance along (011)", format="%.3", $
        fill = all_ones
HDF_SD_ADDDATA, Dsid(Did), computeda
Did = Did + 1
Dsid(Did) = HDF_SD_CREATE(sfid, "ic1", /LONG, dims)
HDF_SD_ATTRSET, Dsid(Did), "gain", gain1
HDF_SD_ATTRSET, Dsid(Did), "I_monitor", " "
HDF_SD_SETINFO, Dsid(Did), coordsys="monitor", unit="photons", format="%.5", $
        fill = all_ones
HDF_SD_ADDDATA, Dsid(Did), ic1a
```

```
Did = Did + 1
Dsid(Did) = HDF_SD_CREATE(sfid, "scint", /LONG, dims)
HDF_SD_ATTRSET, Dsid(Did), "signal", 1
HDF_SD_ATTRSET, Dsid(Did), "primary", 1
HDF_SD_SETINFO, Dsid(Did), coordsys="detector", unit="photons", format="%.5", $
        fill = all_ones
HDF_SD_ADDDATA, Dsid(Did), scint

entry_log, 5, vgN, vgD, id, Did, sid, Dsid
for i=0, Did do HDF_VG_ADDTR, vgD, DFTAG_NDG, HDF_SD_IDTOREF(Dsid(i))
for i=0, Did do HDF_SD_ENDACCESS, Dsid(i)
HDF_VG_DETACH, vgD
for i=0, id do HDF_VG_ADDTR, vgN, DFTAG_NDG, HDF_SD_IDTOREF(sid(i))
HDF_VG_ADDTR, vgN, DFTAG_VG, vgD
for i=0, id do HDF_SD_ENDACCESS, sid(i)
HDF_VG_DETACH, vgN

; entry6,  A circle in hkl space, with points every degree.
print
print, "starting 'entry6'"
; make a dummy scan of 360 points
x = findgen(360) * !pi/180.
h1 = .1 * cos(x)
k1 = .1 * sin(x)
l1 = fltarr(360) + 2.
ic1 = lonarr(360) + 200000L
scint = long(sin(x)*10)
vgN = HDF_VG_ATTACH(vfid, -1, /WRITE)
HDF_VG_SETINFO, vgN, name="entry6", class="APS_entry"
id = 0
sid(id) = sdstring(sfid,"date",'25-Feb-1996')
id = id + 1
sid(id) = sdstring(sfid,"hour",'02:10:59.10 -7')
id = id + 1
sid(id) = sdstring(sfid,"entry_analysis","diffraction/misc")
id = id + 1
sid(id) = sdstring(sfid,"entry_intent","data")
id = id + 1
sid(id) = sdstring(sfid,"title",'0.1 rad circle about (002)')
id = id + 1
sid(id) = sdstring(sfid,"constraint",$
      "|(hkl)-(002)|=.1 and (hkl)-(002) perpendicular to (002)")

mono_energy = 8.047
id = id + 1
sid(id) = HDF_SD_CREATE(sfid, "mono_energy", /FLOAT, [1])
HDF_SD_SETINFO, sid(id), coordsys="energy", unit="keV", format="%.4"
HDF_SD_ADDDATA, sid(id), mono_energy
```

```
hkl_near=float([0,0,2])
dims=[3]
id = id + 1
sid(id) = HDF_SD_CREATE(sfid, "hkl_near", /FLOAT, dims)
HDF_SD_ADDDATA, sid(id), hkl_near
vgD = HDF_VG_ATTACH(vfid, -1, /WRITE)
HDF_VG_SETINFO, vgD, name="data1", class="APS_scan"
Did = 0
dims=[360]

Dsid(Did) = HDF_SD_CREATE(sfid, "h1", /FLOAT, dims)
HDF_SD_SETINFO, Dsid(Did), coordsys="h", format="%.3"
; diffract_axis set to h1 in entry_default
HDF_SD_ADDDATA, Dsid(Did), h1
Did = Did + 1
Dsid(Did) = HDF_SD_CREATE(sfid, "k1", /FLOAT, dims)
HDF_SD_SETINFO, Dsid(Did), coordsys="k", format="%.3"
; diffract_axis set to k1 in entry_default
HDF_SD_ADDDATA, Dsid(Did), k1
Did = Did + 1
Dsid(Did) = HDF_SD_CREATE(sfid, "l1", /FLOAT, dims)
HDF_SD_SETINFO, Dsid(Did), coordsys="l", format="%.3"
; diffract_axis set to l1 in entry_default
HDF_SD_ADDDATA, Dsid(Did), l1

computed = findgen(dims(0))
Did = Did + 1
Dsid(Did) = HDF_SD_CREATE(sfid, "circle", /FLOAT, dims)
equation="angle"
HDF_SD_ATTRSET, Dsid(Did), "equation", equation
HDF_SD_ATTRSET, Dsid(Did), "axis", 1
HDF_SD_SETINFO, Dsid(Did), coordsys="angle on circle", unit="degrees", $
        format="%.1", fill=all_ones
HDF_SD_ADDDATA, Dsid(Did), computed
Did = Did + 1
Dsid(Did) = HDF_SD_CREATE(sfid, "ic1", /LONG, dims)
HDF_SD_ATTRSET, Dsid(Did), "gain", gain1
HDF_SD_ATTRSET, Dsid(Did), "I_monitor", " "
HDF_SD_SETINFO, Dsid(Did), coordsys="monitor", unit="photons", format="%.5", $
        fill=all_ones
HDF_SD_ADDDATA, Dsid(Did), ic1
Did = Did + 1
Dsid(Did) = HDF_SD_CREATE(sfid, "scint", /LONG, dims)
HDF_SD_ATTRSET, Dsid(Did), "signal", 1
HDF_SD_ATTRSET, Dsid(Did), "primary", 1
HDF_SD_SETINFO, Dsid(Did), coordsys="detector", unit="photons", format="%.5", $
        fill=all_ones
```

```
HDF_SD_ADDDATA, Dsid(Did), scint

entry_log, 6, vgN, vgD, id, Did, sid, Dsid
for i=0, Did do HDF_VG_ADDTR, vgD, DFTAG_NDG, HDF_SD_IDTOREF(Dsid(i))
for i=0, Did do HDF_SD_ENDACCESS, Dsid(i)
HDF_VG_DETACH, vgD
for i=0, id do HDF_VG_ADDTR, vgN, DFTAG_NDG, HDF_SD_IDTOREF(sid(i))
HDF_VG_ADDTR, vgN, DFTAG_VG, vgD
for i=0, id do HDF_SD_ENDACCESS, sid(i)
HDF_VG_DETACH, vgN

print
print, "calling end routines and closing the file"
HDF_SD_END, sfid
HDF_CLOSE, vfid
;print, format='(a)', BELL
end


 ********************************************************** ******************

;APS_HDF_BROWSER.PRO

pro build_vgroup_menu, state, vgn, desc, flag
  HDF_VG_GETINFO, vgn, nentries=n, name=name
  HDF_VG_GETTRS, vgn, tags, refs
  if (state.view_mode eq 0) then $
    select=where(tags eq 1965, n) else $
    select=where((tags eq 720) or (tags eq 1965), n)
  if (n eq 0) then flag=flag-1
  desc = [desc, {CW_PD_MENU_S, flags:flag, name:name}]
  for i=0, n-1 do begin
    ; If this is the last item in this Vgroup set flag=3
    if (i eq n-1) then flag=3 else flag=1
    j = select(i)
    case tags(j) of
      720: begin
        index = hdf_sd_reftoindex(state.sdsid, refs(j))
        sds_id = hdf_sd_select(state.sdsid, index)
        build_sds_menu, state, sds_id, desc, flag
      end
      1965: begin
        t = HDF_VG_ATTACH(state.fileid, refs(j))
        build_vgroup_menu, state, t, desc, flag
      end
    endcase
  endfor
end
```

```
pro build_sds_menu, state, sds_id, desc, flag
  HDF_SD_GETINFO, sds_id, name=name, dims=dims, type=type, natts=natts
  if ((type eq 'STRING') and (n_elements(dims) eq 1)) then begin
    HDF_SD_GETDATA, sds_id, data
    str = name + '  ' + ' = "' + data(0) + ""
  endif else if ((n_elements(dims) eq 1) and (dims(0) eq 1)) then begin
    HDF_SD_GETDATA, sds_id, data
    str = name + '  ' + ' = ' + strtrim(string(data(0)),2)
  endif else begin
    count = dims*0 + 1 & count(0)=2
    HDF_SD_GETDATA, sds_id, data, count=count
    str = name + '  ' + type + '( '
    for i=0, n_elements(dims)-1 do str = str+strtrim(dims(i),2)+' '
    str = str + ')'
    str = str + ' = ' + strtrim(string(data(0)),2) + ', ' + $
                        strtrim(string(data(1)),2) + ', ...'
  endelse
  if (state.view_mode lt 2) then natts = 0  ; Don't display attributes
  if (natts eq 0) then flag=flag-1
  desc = [desc, {CW_PD_MENU_S, flags:flag, name:str}]
  if (natts gt 0) then begin
    for j=0,natts-1 do begin
       HDF_SD_ATTRINFO,sds_id,j,name=name,data=d,count=count,type=t ype
      if (type eq 'STRING') then begin
        d = "" + d + ""
      endif
      str = name+' = '+string(d)
      desc = [desc, {CW_PD_MENU_S, flags:0, name:str(0)}]
    endfor
    desc(n_elements(desc)-1).flags=2
  endif
  HDF_SD_ENDACCESS,sds_id
end

pro build_globatts_menu, state, desc
  hdf_sd_fileinfo,state.sdsid,nmfsds,nglobatts
  if (nglobatts gt 0) then begin
    desc = [desc, {CW_PD_MENU_S, flags:1, name:'Global attributes'}]
    for j=0,nglobatts-1 do begin
       HDF_SD_ATTRINFO,state.sdsid,j,name=name,data=d,count=count,t ype=type
      if (type eq 'STRING') then begin
        d = "" + d + ""
      endif
      str = name+' = '+string(d)
      desc = [desc, {CW_PD_MENU_S, flags:0, name:str(0)}]
    endfor
```

```
    desc(n_elements(desc)-1).flags=2
  endif
end


pro aps_build_menu, state, desc
; Loop on Vgroups of class APS_entry
desc = {CW_PD_MENU_S, flags:1, name:'Browse ...'}
if (state.fileid eq -1) then return
numvg=hdf_number(state.fileid,tag=1965)
vgid = HDF_VG_GETID(state.fileid, -1)
build_globatts_menu, state, desc
for i=0, numvg-1 do begin
  vgn = HDF_VG_ATTACH(state.fileid, vgid)
  HDF_VG_GETINFO, vgn, class=class, name=name
  if  (class eq 'APS_entry') or (class eq 'APS_default') then begin
    build_vgroup_menu, state, vgn, desc, 1
  endif
  vgid = HDF_VG_GETID(state.fileid, vgid)
endfor
end


pro aps_open_file, state
  ;   Check that it is HDF
if (state.filename ne "") then begin
;  if hdf_ishdf(state.filename) ne 1 then begin
; TEMPORARY TEST
   if (1 ne 1) then begin
    status = widget_message(/ERROR, $
       "File "+state.filename+" is not an HDF file.")
    state.fileid=-1
    state.sdsid=-1
   endif else begin
     state.fileid=hdf_open(state.filename,/read)
     state.sdsid=hdf_sd_start(state.filename,/read)
   endelse
endif else begin
    state.fileid=-1
    state.sdsid=-1
endelse
end

pro new_menu, state
    aps_open_file, state
    aps_build_menu, state, menu
    ;for i=0, n_elements(menu)-1 do print, menu(i)
    state.widget.view = $
```

```
      cw_pdmenu(state.widget.view_parent, menu, /return_index)
   if (state.fileid eq -1) then $
      widget_control, state.widget.view, sensitive=0 else $
      widget_control, state.widget.view, sensitive=1
end

pro aps_hdf_browser_event, event

widget_control, event.top, get_uvalue=state, /no_copy

case event.id of

   state.widget.open: begin
      state.filename = pickfile()
      widget_control, event.top, update=0, /hourglass
      widget_control, state.widget.view, /destroy
      new_menu, state
      widget_control, event.top, update=1
   end

   state.widget.exit: begin
      widget_control, event.top, /destroy
      return
   end

   state.widget.view_mode: begin
      ; Don't do anything if a new selection was not made
      if (state.view_mode ne event.index) then begin
         state.view_mode = event.index
          widget_control, event.top, update=0, /hourglass
          widget_control, state.widget.view, /destroy
          new_menu, state
          widget_control, event.top, update=1
      endif
   end

   else: begin
      ; This must be a pull-down menu event
      ; print, 'Event value = ', event.value
   end
endcase
widget_control, event.top, set_uvalue=state, /no_copy
end


pro aps_hdf_browser, filename

widget = { $
```

```
          open:       0L, $
          close:      0L, $
          exit:       0L, $
          view_parent:0L, $
          view_mode:  0L, $
          view:       0L $
}

state = { $
       widget:     widget, $
       filename:   "", $
       sdsid:      -1L, $
       view_mode:  2L, $
       fileid:     -1L}

base = widget_base( title = 'APS HDF Browser', mbar = mbar, /column, $
                  resource = 'aps_hdf_browser')

default_font = get_font_name(/medium, /bold)
large_font = get_font_name(/large, /italic, /bold)
widget_control, base, default_font = default_font

file          = widget_button( mbar, /menu,              $
                           font = large_font, $
                           value = 'File      ')
state.widget.open          = widget_button( file, value = 'Open . . .')
state.widget.close         = widget_button( file, value = 'Close')
state.widget.exit          = widget_button( file, /separator, value = 'Exit',$
                                  font = large_font)

state.widget.view_parent = widget_base( base, /column)
state.widget.view_mode = widget_droplist(state.widget.view_parent, $
   value=['Vgroups only', 'Vgroups and SDS', 'Vgroups, SDS and attributes'])
widget_control, state.widget.view_mode, set_droplist_select=state.view_mode
if (n_elements(filename) ne 0) then state.filename = filename
aps_open_file, state
new_menu, state
widget_control, base, set_uvalue=state
widget_control, base, /realize
xmanager, 'aps_hdf_browser', base
end
```