Wizard Hooks in GeoPIXE

Command in GeoPIXE can be accessed using hooks using the Notify mechanism, which send an event to the destination window. A broadcast Notify is used, which means that any window of the type requested will receive and act on the request. This keeps the mechanism easy to use. However, for this reason, make sure you only have one of each window open (e.g. no Image window clones), as all would act on a command.

Contents

Wizard Hooks in GeoPIXE 1

Window Names 2
Hooks by Window 2
Sort EVT 2
Image 3
Spectrum Display 4
Image Regions 7
Image RGB 7
Spectrum Fit 8

Hooks in GeoPIXE are accessed by sending a broadcast Notify message in IDL to 'wizard-notify' (see example code for 'Wizard_Depth.pro') and pass this structure (example for sorting an image):

```
{ wizard:     'standards', $          ; name of originating Wizard
window:       'Sort EVT', $           ; name of destination Window in GeoPIXE
command:      'sort-image', $         ; command to be executed by the Window
qual1:        '', $                   ; qualifier #1 to command
qual2:        '', $                   ; qualifier #2 to command
error:        0, $                    ; a return error code (0=good, 1=error)
pdata:        ptr_new( {...}), $      ; a passed data structure ptr, specific to command
local:        1, $                    ; 'pdata' managed (e.g. freed) by Wizard
callback:     'routine', $            ; routine to execute in Wizard on return
pnext:        pnext }                 ; pointer to the next 'wizard_notify' struct
```

The structure contains the name of the destination window in GeoPIXE (see the list below under 'Window Names') and the name of the command to execute there, plus some qualifiers and a pointer to a data struct specific to this command. The struct can also point to another 'wizard_notify' struct to form a linked list. In this case each is executed in turn, managed by the Wizard (see example code for 'Wizard_Depth.pro').

This struct can be defined using a call to the 'define' function:

```
p = define(/wizard_notify)
```

If 'pdata' is just to be used for some return value, and not passing a parameter/struct, then remember to use ptr_new(/allocate_heap), or use an established pointer storage variable, to provide a pointer to valid heap storage for the return value.

Generally, it is the responsibility of the Wizard to allocate and free heap memory pointed to by these pointers. If 'local'=0, then this means that 'pdata' is a pointer managed by another window. For example, if a hook wants to pass a copy of a large image pointer back to a Wizard, without making a separate copy of the whole image struct, then it can set local=0 to signify that the other window maintains responsibility for this

memory (e.g. to free it). In this case, the Wizard should NOT free this ‘pdata’ memory. Always free this struct using the command: “free_wizard, p”. ‘free_wizard’ will not free ‘pdata’ if ‘local’ is zero.

Window Names

Sort EVT	Sort EVT window
Image	main Image window (should have only 1 open)
Image Regions	Image Regions window
Image RGB	RGB Images window
Spectrum Display	spectrum display window
Spectrum Fit	X-ray Spectrum fitting window

Hooks by Window

In addition to the ‘command’ codes list below, all windows should respond to the command ‘open-test’, which simply returns with error=0 if the window is open on the screen. This is used by each Wizard to probe the existence of necessary windows.

Sort EVT

sort-image

Sets requested parameters in the Sort EVT window, and then does the sort.

```
{ wizard:     ‘?’, $                ; name of originating Wizard
window:       ‘Sort EVT’, $         ; name of destination Window in GeoPIXE
command:      ‘sort-image’, $       ; command to be executed by the Window
qual1:        ‘string1’, $          ; add this tag to the end of the output file-name
qual2:        ‘string2’, $          ; remove this tag from end of the output file-name
error:        0, $                  ; a return error code (0=good, 1=error)
pdata:        ptr_new( {…}), $      ; a passed data structure ptr, specific to command
local:        1, $                  ; ‘pdata’ managed (e.g. freed) by Wizard
callback:     ‘routine’, $          ; routine to execute in Wizard on return
pnext:        pnext }               ; pointer to the next ‘wizard_notify’ struct
```

A full ‘pdata’ pointer looks like this (unneeded items can be omitted for this data struct):

```
pdata = ptr_new( {      $
        device:         ‘MAIA_DEVIC E’, $   ; device name
        image_mode:     0, $                ; sort mode (0=images)
        type:           7, $                ; data type (0=PIXE, 7=SXRF)
        array:          1, $                ; array detector (1=array, 0=single)
        blog:           ‘file1’, $          ; (first) raw data file
        pileup:         ‘pileup’, $         ; pileup file
        throttle:       ‘throttle’, $       ; throttle file
        output:         ‘output’, $         ; output file name
        verify:         1, $                ; file-name verification (enable=1, disable=0)
        pnew:           pnew, $             ; if /verify, return old and new file-names
        conv:           1.0, $              ; initial 'conv' conversion factor from IC to charge
        charge_mode:    1, $                ; flux/charge mode (0=charge, 1=IC,PV, 2=no PV)
        flux_scaler:    'Maia:scaler.FC0', $ ; scaler ID name
        gain_value:     gain, $             ; IC gain
        gain_units:     1.0, $              ; scales gain into units of 'nA/V'
```

```
cal:          'energy_cal_file', $     ; energy calibration/ detector selection file
proj_mode:    'DA', $                  ; projection mode ('DA', 'CUTS','STIM')
dam:          'da_file' }, $           ; DA matrix file name
/no_copy)
```

If the text "-string2" is found at the end of the output file-name in Sort EVT, it will be stripped off and the string "-string1" appended (if not already present).

If any entry is missing from 'pdata', then that parameter is not set and the current value in Sort EVT is used. Hence, a simple 'sort-image' request struct (as used in 'Wizard_Depth') may just have one or two items to set one or two parameters. However, that means that the user must set-up Sort EVT beforehand. In contrast, the 'Wizard_Standards' Wizard sets most of these parameters.

If file verification is requested (verify=1) in 'pdata', then the 'pnew' pointer (should be created /allocate_heap) returns a struct of old and new file-names: {blog.blog, pileup:pileup, throttle:throttle, dam:dam}, where each parameter has a 'new' and 'old' file-name string entry (e.g. pileup={old:'', new:''}).

Image

load-image

Loads an image file into the Image window.

```
{ wizard:     '?', $                   ; name of originating Wizard
window:       'Image', $               ; name of destination Window in GeoPIXE
command:      'load-image', $          ; command to be executed by the Window
error:        0, $                     ; a return error code (0=good, 1=error)
pdata:        ptr_new( ), $            ; pointer to file name to load
local:        1, $                     ; 'pdata' managed (e.g. freed) by Wizard
callback:     'routine', $             ; routine to execute in Wizard on return
pnext:        pnext }                  ; pointer to the next 'wizard_notify' struct
```

In this case the 'pdata' pointer simply points to a file-name string.

sum-region

Sets a region shape on the image and integrates it, and returns a pointer to the Region results struct. See 'define(/table)' for a region results struct, which holds the results for a row of the Image Regions window.

```
{ wizard:     '?', $                   ; name of originating Wizard
window:       'Image', $               ; name of destination Window in GeoPIXE
command:      'sum-region, $           ; command to be executed by the Window
error:        0, $                     ; a return error code (0=good, 1=error)
pdata:        ptr_new( ), $            ; pointer to region specification struct
local:        1, $                     ; 'pdata' managed (e.g. freed) by Wizard
callback:     'routine', $             ; routine to execute in Wizard on return
pnext:        pnext }                  ; pointer to the next 'wizard_notify' struct
```

The 'pdata' pointer looks like this:

```
{ mode:       0, $                     ; "+" sum mode (0="+", 1='-')
shape:        1, $                     ; Box shape index (1=Box, 2=Circle, 3=Curve8, …)
x:            x, $                     ; X handles
y:            y, $                     ; Y handles
theta:        theta, $                 ; Rotation angle
```

```
presults:       pregion, $                  ; Region results struct
local:          0 }                         ; 'presults' not managed (e.g. freed) by Wizard
```

Spectrum Display

import-spectra

Load spectra from a raw data file into the Spectrum Display window.

```
{ wizard:       '?', $                      ; name of originating Wizard
window:         'Spectrum Display', $       ; name of destination Window in GeoPIXE
command:        'import-spectra', $         ; command to be executed by the Window
error:          0, $                        ; a return error code (0=good, 1=error)
pdata:          ptr_new( ), $               ; pointer to import options struct
local:          1, $                        ; 'pdata' managed (e.g. freed) by Wizard
callback:       'routine', $                ; routine to execute in Wizard on return
pnext:          pnext }                     ; pointer to the next 'wizard_notify' struct
```

The 'pdata' pointer looks like this:

```
{ blog:         blog, $                     ; blog file
opt:            opt, $                      ; import options struct (from 'import_select')
pileup:         pileup, $                   ; pileup file
throttle:       throttle, $                 ; throttle file
linear:         linear, $                   ; optional re-linearization file
output:         output_dir, $               ; output path
verify:         1, $                        ; file-name verification (enable=1, disable=0)
pnew:           ptr_new(/allocate_heap), $  ; if /verify, return old and new file-names
charge_mode:    charge_mode, $              ; flux/charge mode
flux_scaler:    pv, $                       ; scaler ID
sensitivity:    gain, $                     ; IC gain
conv:           conv, $                     ; 'conv' factor
charge:         charge }                    ; charge (uC) (passed, updated returned too)
```

The 'opt' struct is returned by the 'import_select' and depends on the file format selected from the device list (from 'get_import_list' routine and defined using "define(/import)" routine). It takes this form:

```
{ name:         '', $                       ; unique name for import
title:          '', $                       ; description for import list
in_ext:         '.evt', $                   ; input file extension
request:        title, $                    ; title for file requester
preview:        0, $                        ; allow spectrum preview
raw:            0, $                        ; flags use of separate Raw data path
spec_evt:       0, $                        ; uses a call to 'spec_evt' to extract from EVT data
device_name:    '', $                       ; associated device object name
multifile:      0, $                        ; denotes data in a series of more than one file
separate:       '.', $                      ; char between file and run #
use_linear:     0, $                        ; request linearization file
use_pileup:     0, $                        ; request pileup file
use_throttle:   0, $                        ; request throttle file
use_IC:         0, $                        ; needs a PV selection
IC_mode:        0, $                        ; default charge/flux mode
use_tot:        0, $                        ; collect ToT data too
xr:             200L, $                     ; default X range
yr:             200L }                      ; default Y range
```

If file verification is requested (verify=1) in 'pdata', then the 'pnew' pointer (should be created /allocate_heap) returns a struct of old and new file-names: { pileup:pileup, throttle:throttle, linear:linear}, where each parameter has a 'new' and 'old' file-name string entry (e.g. pileup={old:'', new:''}).

load-spectra

Load spectra from SPEC file into the Spectrum Display window.

```
{ wizard:     '?', $                    ; name of originating Wizard
window:       'Spectrum Display', $     ; name of destination Window in GeoPIXE
command:      'load-spectra', $         ; command to be executed by the Window
error:        0, $                      ; a return error code (0=good, 1=error)
pdata:        ptr_new( ), $             ; pointer to file name(s) to load
local:        1, $                      ; 'pdata' managed (e.g. freed) by Wizard
callback:     'routine', $              ; routine to execute in Wizard on return
pnext:        pnext }                   ; pointer to the next 'wizard_notify' struct
```

In this case the 'pdata' pointer simply points to a file-name string, or an array of strings for multiple file load.

save-spectra

Sums all detector spectra in *Spectrum Display*.

```
{ wizard:     '?', $                    ; name of originating Wizard
window:       'Image', $                ; name of destination Window in GeoPIXE
command:      'save-spectra', $         ; command to be executed by the Window
error:        0, $                      ; a return error code (0=good, 1=error)
pdata:        ptr_new( ), $             ; pointer to filename string
local:        1, $                      ; 'pdata' managed (e.g. freed) by Wizard
callback:     'routine', $              ; routine to execute in Wizard on return
pnext:        pnext }                   ; pointer to the next 'wizard_notify' struct
```

In this case the 'pdata' pointer simply points to a SPEC file-name string.

energy-cal

Loads a selected energy-calibration SPEC file and sets the energy calibration of all spectra based on detector number.

```
{ wizard:     '?', $                    ; name of originating Wizard
window:       'Image', $                ; name of destination Window in GeoPIXE
command:      'energy-cal, $            ; command to be executed by the Window
error:        0, $                      ; a return error code (0=good, 1=error)
pdata:        ptr_new( ), $             ; pointer to an energy cal SPEC filename string
local:        1, $                      ; 'pdata' managed (e.g. freed) by Wizard
callback:     'routine', $              ; routine to execute in Wizard on return
pnext:        pnext }                   ; pointer to the next 'wizard_notify' struct
```

In this case the 'pdata' pointer simply points to a SPEC file-name string.

sum-spectra

Sums all detector spectra in *Spectrum Display*.

```
{ wizard:     '?', $                    ; name of originating Wizard
window:       'Image', $                ; name of destination Window in GeoPIXE
```

```
command:    'sum-spectra', $          ; command to be executed by the Window
error:      0, $                      ; a return error code (0=good, 1=error)
callback:   'routine', $              ; routine to execute in Wizard on return
pnext:      pnext }                   ; pointer to the next 'wizard_notify' struct
```

The 'pdata' pointer looks like this:

```
{ select:   'select-file', $          ; select (*.select.csv) or spec (*.spec) file
label:      'label', $                ; 'Label' string for spec header
poutput:    poutput }                 ; pointer to output file-name
```

sum-selected-spectra

Sums selected detector spectra, selected using either a selection file (*.select.csv) or using the detector channels present in a spectrum file (*.spec).

```
{ wizard:   '?', $                    ; name of originating Wizard
window:     'Image', $                ; name of destination Window in GeoPIXE
command:    'sum-selected-spectra', $ ; command to be executed by the Window
error:      0, $                      ; a return error code (0=good, 1=error)
pdata:      ptr_new( ), $             ; pointer to the selection and output
local:      1, $                      ; 'pdata' managed (e.g. freed) by Wizard
callback:   'routine', $              ; routine to execute in Wizard on return
pnext:      pnext }                   ; pointer to the next 'wizard_notify' struct
```

The 'pdata' pointer looks like this:

```
{ select:   'select-file', $          ; select (*.select.csv) or spec (*.spec) file
label:      'label', $                ; 'Label' string for spec header
poutput:    poutput }                 ; pointer to output file-name
```

select-spectra

Use a detector channel selection file (*.select.csv; or the channels present in a *.spec file) to select detector channels in Spectrum Display and then write out the selected channels to a spec file.

```
{ wizard:   '?', $                    ; name of originating Wizard
window:     'Spectrum Display', $     ; name of destination Window in GeoPIXE
command:    'select-spectra', $       ; command to be executed by the Window
error:      0, $                      ; a return error code (0=good, 1=error)
pdata:      ptr_new( ), $             ; pointer to the selection and output
local:      1, $                      ; 'pdata' managed (e.g. freed) by Wizard
callback:   'routine', $              ; routine to execute in Wizard on return
pnext:      pnext }                   ; pointer to the next 'wizard_notify' struct
```

The 'pdata' pointer looks like this:

```
{ select:   'select-file', $          ; select (*.select.csv) or spec (*.spec) file
poutput:    poutput }                 ; ptr to output file-name name
```

outer-inner-ratio

Must follow fitting of the outer and inner spectra. Forms outer/inner ratio of fit results for the selected element and returns an array of outer and inner peak areas and error estimates: [area_outer, err_outer, area_inner, err_inner] pointed to by 'pdata'.

```
{ wizard:   '?', $                    ; name of originating Wizard
window:     'Spectrum Display', $     ; name of destination Window in GeoPIXE
```

```
command:    'outer-inner-ratio', $          ; command to be executed by the Window
qual1:      el, $                           ; element name string
error:      0, $                            ; a return error code (0=good, 1=error)
pdata:      ptr_new(/allocate_heap ), $     ; pointer to the result array returned
local:      1, $                            ; 'pdata' managed (e.g. freed) by Wizard
callback:   'routine', $                    ; routine to execute in Wizard on return
pnext:      pnext }                         ; pointer to the next 'wizard_notify' struct
```

The 'pdata' pointer looks like this:

[area_outer, err_outer, area_inner, err_inner]

Remember to /allocate_heap for 'pdata' (or use an established pointer to heap), so a return value can use the same storage.

Note that this command must be preceded by fitting of the outer and inner sum spectra, respectively. This is used in 'Wizard_Depth'.

Image Regions

extract-individual

Extract spectra for the individual detector channels using the current region #0 in the Image Regions window.

```
{ wizard:   '?', $                      ; name of originating Wizard
window:     'Image Regions', $          ; name of destination Window in GeoPIXE
command:    'extract-individual', $     ; command to be executed by the Window
error:      0, $                        ; a return error code (0=good, 1=error)
pdata:      ptr_new(/allocate_heap ), $ ; pointer to returned spec file name
local:      1, $                        ; 'pdata' managed (e.g. freed) by Wizard
callback:   'routine', $                ; routine to execute in Wizard on return
pnext:      pnext }                     ; pointer to the next 'wizard_notify' struct
```

In this case the 'pdata' pointer on return simply points to a file-name string for the output SPEC file. Remember to /allocate_heap for this one (or use an established pointer to heap), so a return value can use the same storage.

This command assumes that the regions exist and have been saved to (or read from) a regions file. It returns the resulting spec file name.

Image RGB

compare-image

Must follow the loading of a valid Inner image data-set into the Image window. This then loads a second image data-set for comparison, which makes both the original and this latter 'compare' set of elements available on the element droplists in the Image RGB window.

```
{ wizard:   '?', $                      ; name of originating Wizard
window:     'Spectrum Display', $       ; name of destination Window in GeoPIXE
command:    'outer-inner-ratio', $      ; command to be executed by the Window
qual1:      el, $                       ; element name string
error:      0, $                        ; a return error code (0=good, 1=error)
pdata:      ptr_new( ), $               ; pointer to outer image DAI file name
```

```
local:      1, $                    ; 'pdata' managed (e.g. freed) by Wizard
callback:   'routine', $            ; routine to execute in Wizard on return
pnext:      pnext }                 ; pointer to the next 'wizard_notify' struct
```

In this case the 'pdata' pointer simply points to a file-name string for the outer image DAI file name.

Note that this command must follow the loading of a valid Inner image data-set into the Image window.

Spectrum Fit

fit-one

Execute a "Fit All" on the X-ray Spectrum Fit window. It assumes that spectra exist and the fit window has been set-up previously.

```
{ wizard:   '?', $                  ; name of originating Wizard
window:     'Spectrum Fit', $       ; name of destination Window in GeoPIXE
command:    'fit-all', $            ; command to be executed by the Window
error:      0, $                    ; a return error code (0=good, 1=error)
pdata:      ptr_new( ), $           ; pointer to array of pointers to fitting results structs
local:      0, $                    ; 'pdata' managed (e.g. freed) by GeoPIXE
callback:   'routine', $            ; routine to execute in Wizard on return
pnext:      pnext }                 ; pointer to the next 'wizard_notify' struct
```

The 'pdata' pointer looks like this:

```
{ presults:   ptr_new() }           ; points to array of pointers to fitting results structs
                                    ; which remain managed by GeoPIXE
```

fit-all

Execute a "Fit All" on the X-ray Spectrum Fit window. It assumes that spectra exist and the fit window has been set-up previously.

```
{ wizard:   '?', $                  ; name of originating Wizard
window:     'Spectrum Fit', $       ; name of destination Window in GeoPIXE
command:    'fit-all', $            ; command to be executed by the Window
error:      0, $                    ; a return error code (0=good, 1=error)
callback:   'routine', $            ; routine to execute in Wizard on return
pnext:      pnext }                 ; pointer to the next 'wizard_notify' struct
```

Spectrum Select

delete

Execute a "Fit All" on the X-ray Spectrum Fit window. It assumes that spectra exist and the fit window has been set-up previously.

```
{ wizard:   '?', $                  ; name of originating Wizard
window:     'Spectrum Select, $     ; name of destination Window in GeoPIXE
command:    'delete, $              ; command to be executed by the Window
error:      0, $                    ; a return error code (0=good, 1=error)
pdata:      ptr_new( ), $           ; pointer to a selector string
local:      1, $                    ; 'pdata' managed (e.g. freed) by Wizard
callback:   'routine', $            ; routine to execute in Wizard on return
pnext:      pnext }                 ; pointer to the next 'wizard_notify' struct
```

In this case the ‘pdata’ pointer simply points to a string that specifies what to delete (like the delete droplist selector on the *Spectrum Select* window):

xyt	XY, T spectra
xy	XY spectra
t	T spectra
e	E spectra
odd	odd detector channels
even	even detector channels
selected	selected spectra
displayed	displayed spectra
overlays	fit and other overlays on selected spectra
charge	charge values in spectrum struct
not-selected	spectra not selected
not-displayed	spectra not displayed
invalid-cal	spectra with invalid energy cal (A=1, B=0)