areaDetector

Writing Plugins

Mark Rivers

University of Chicago

Advanced Photon Source

Existing Plugins

Plugin	Generates new NDArrays	Limitations
NDPluginROI	Yes	None
NDPluginStats	No	Centroid, Profiles limited to 2-D mono
NDPluginProcess	Yes	None
NDPluginTransform	Yes	2-D mono or color only
NDPluginStdArrays	No	None
NDPluginOverlay	Yes	2-D mono or color only
NDPluginROIStat	No	None
NDPluginColorConvert	Yes	2-D mono or color only
NDPluginCircularBuff	Yes	None
NDPluginFile	No	HDF5, netCDF, Nexus: None JPEG, TIFF, Magick: 2-D mono or color only

Guidelines and Rules for Writing an areaDetector Plugin

- Plugins will almost always implement the processCallbacks() function. This function will be called with an NDArray pointer each time an NDArray callback occurs.
  - This function will normally call the NDPluginDriver::processCallbacks() base class function, which handles tasks common to all plugins, including callbacks with information about the array, etc.
- Plugins will generally implement one or more of the writeInt32(), writeFloat64() or writeOctet() functions if they need to act immediately on a new value of a parameter.
  - For many parameters it is normally sufficient to simply have them written to the parameter library, and not to handle them in the writeXXX() functions. The parameters are then retrieved from the parameter library with the getIntParam(), getDoubleParam(), or getStringParam() function calls when they are needed.
- If the writeInt32(), writeFloat64() or writeOctet() functions are implemented they **must** call the base class function for parameters that they do not handle and whose parameter index value is less than the first parameter of this class, i.e. parameters that belong to a base class.

Guidelines and Rules for Writing an areaDetector Plugin

- Plugins will need to call the createParam() function in their constructor if they have additional parameters beyond those in the asynPortDriver or NDPluginDriver base classes.
- Plugins may **never** modify the NDArray that they receive in the processCallbacks() function.
  - The reason is that other plugins may be concurrently operating on the same NDArray, since each is passed the same pointer.
  - This means also that when getting the attributes for this plugin that asynNDArrayDriver::getAttributes(pArray->pAttributeList) must not be called with the NDArray passed to processCallbacks(), because that will modify the NDArray attribute list, and hence the NDArray that other plugins are operating on.
  - Plugins such as NDPluginROI and NDPluginColorConvert create new NDArrays via NDArrayPool::copy() or NDArrayPool::convert() (which copy the attributes to the new array) and then call getAttributes(pArray->pAttributeList) with the new array.
  - The NDPluginFile makes a copy of the attribute list from the original NDArray before calling getAttributes(), but does not need to make a copy of the NDArray because it does not modify it.

Guidelines and Rules for Writing an areaDetector Plugin

- Plugins must release their mutex by calling this->unlock() when they do time-consuming operations.
  - If they do not then they will not be able to queue new NDArrays callbacks or obtain new parameter values.
  - Obviously they must not access memory locations that other threads could modify during this time, so they should only access local variables, not class variables (which includes the parameter library).
- If plugins generate new or modified NDArrays then they must call doCallbacksGenericPointer() so that registered clients can get the values of the new arrays.

Plugin examples

- Look in detail at 3 plugins
  - NDPluginStdArrays
  - NDPluginROI
  - NDPluginStats