Time Source Callback and Attribute Plugin for areaDetector

Garth Brown, Kukhee Kim for Camera Team

U.S. DEPARTMENT OF ENERGY
Office of Science

SLAC NATIONAL ACCELERATOR LABORATORY

Abstract

Timestamp tagging to data and aligning data to individual beam pulse are important parts of data acquisition systems for pulse machines. We have utilized Beam Synchronous Acquisition (BSA) for LCLS, FACET and other accelerator facilities in SLAC. BSA provides a common interface for timestamp tagging and aligning data to beam pulses. Unfortunately, we could not get benefit from the BSA for camera image data due to the image is an array data and it should be stored as image file format. BSA only supports scalar type epics PV. Our timing system in SLAC has 360Hz granularity. Most of image processing and acquiring cycles are non-deterministic in time domain and spend longer time than the 360Hz timestamp update period. These bring difficulty of timestamp tagging for image data.

We have discussed with Mark Rivers, author for the areaDetector module, about time source callback to implement the driver level timestamp tagging with user defined function. The driver level timestamp tagging gives more deterministic and real-time behavior. The timestamp is stored as a part of meta-data of the image and can be utilized by downstream plugins such as file plugin which saves the timestamp into image file format. We also discussed to extend attribute plugin to post out the timestamp related data to epics PVs. He reflected our requirements for the time source callback in the latest areaDetector module and the attribute plugin also.

We are going to describe the details about the time source callback and attribute plugin.

Contents

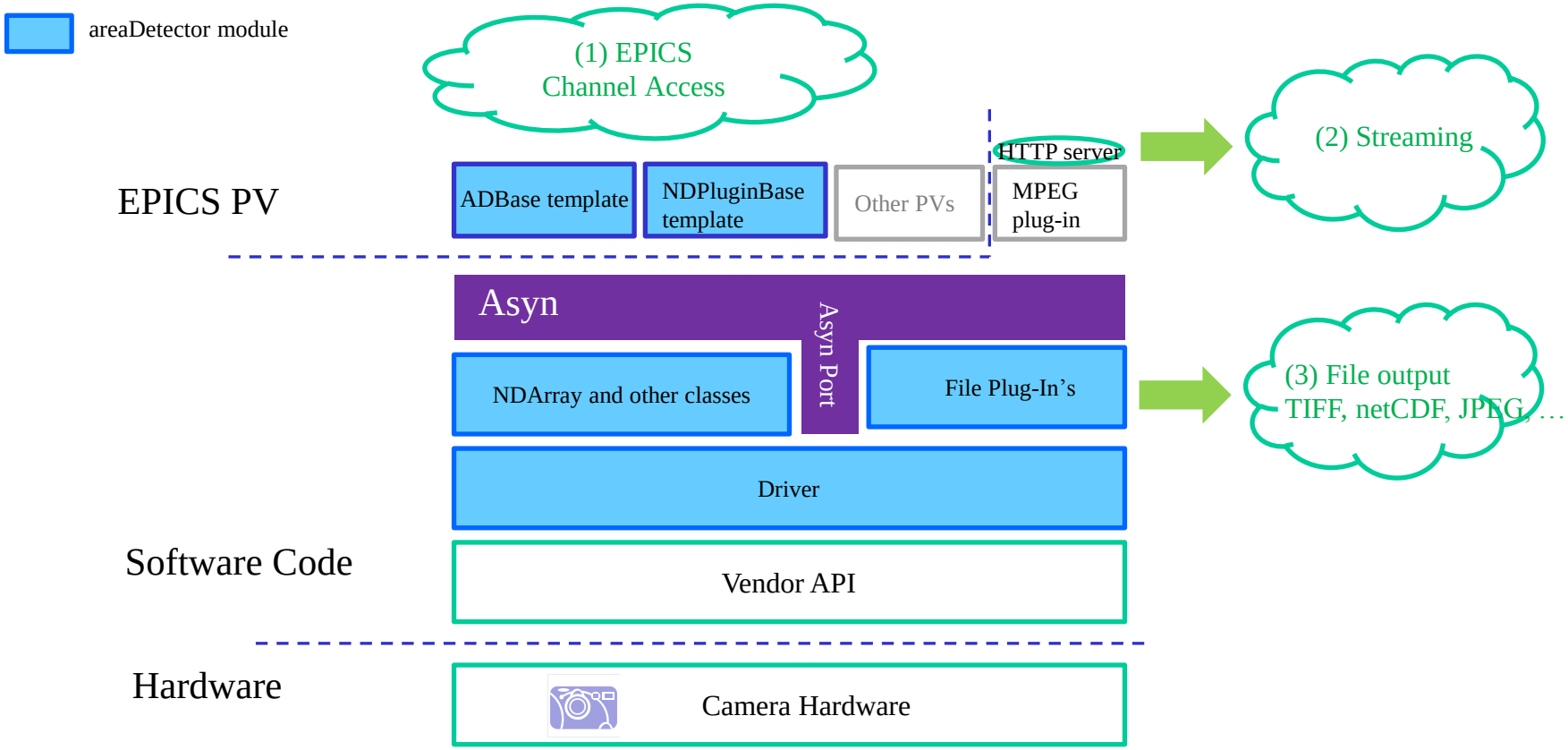
SLAC

- areaDetector
- Time Stamping issues for camera image data
- Improvement of recent areaDetector
  - Time Source callback
  - User Attributes
  - Attributes Plugin
- Conclusion

areaDetector module

SLAC

- A software module to provide general interface for Image Data/Camera in EPICS
  - Not only for Image data, but also for any kind of array
- Witten in C++, work with Asyn



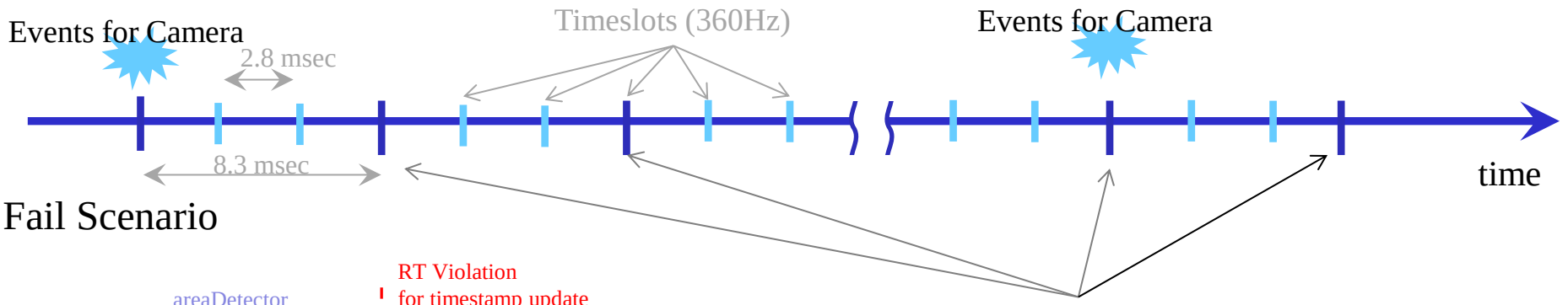
Timestamp Tagging Issue for Image Data

SLAC

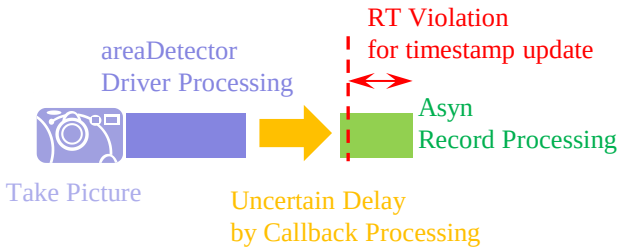
- Image data type (waveform) is not fit for Beam Synchronous Acquisition (BSA) system
- Image acquiring and data processing is not deterministic to fit with 360Hz granularity timing system
- Timestamp/pulse id information should be embedded into a part of image data (meta data in image file)

Timeline for the Timestamp Issue

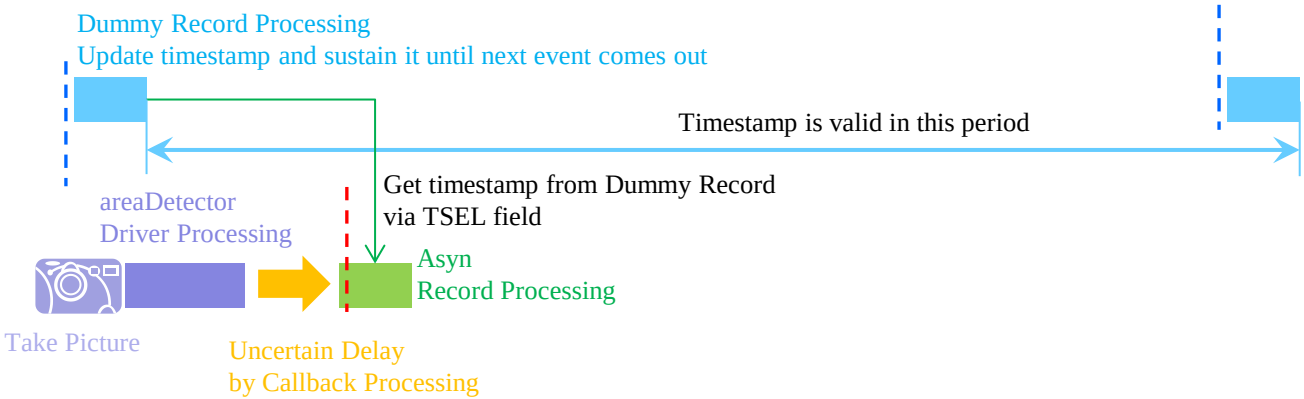
SLAC



Fail Scenario



Temporal Solution



Improvement of areaDetector

SLAC

- Purpose/Goals:

Need to get the Timestamp which includes pulseID supported natively by the areaDetector Package. This needs to be done in such a way that the accelerator community can use this new feature in a generic way. Each facility may different ways in which they supporting timing.

- Closely worked with Mark Rivers and Mark provided improvements for areaDetector
  - Provide hooks to get EPICS timestamp
    - areaDetector already had its own timestamp, double type
    - SLAC pulseID embedded timestamp could not fit into the double type timestamp
  - Provide a callback mechanism to update the epics timestamp which can deliver the SLAC pulseID embedded timestamp into the areaDetector
  - Provide a callback mechanism for a user attribute in the Plug-In's
    - To avoid software change when new attribute is added and new data processing is required for the new attribute
  - Add up epics timestamp (sec, nano-sec) into the virtual attributes in Attribute Plugin to post out to PVs (time series PVs)

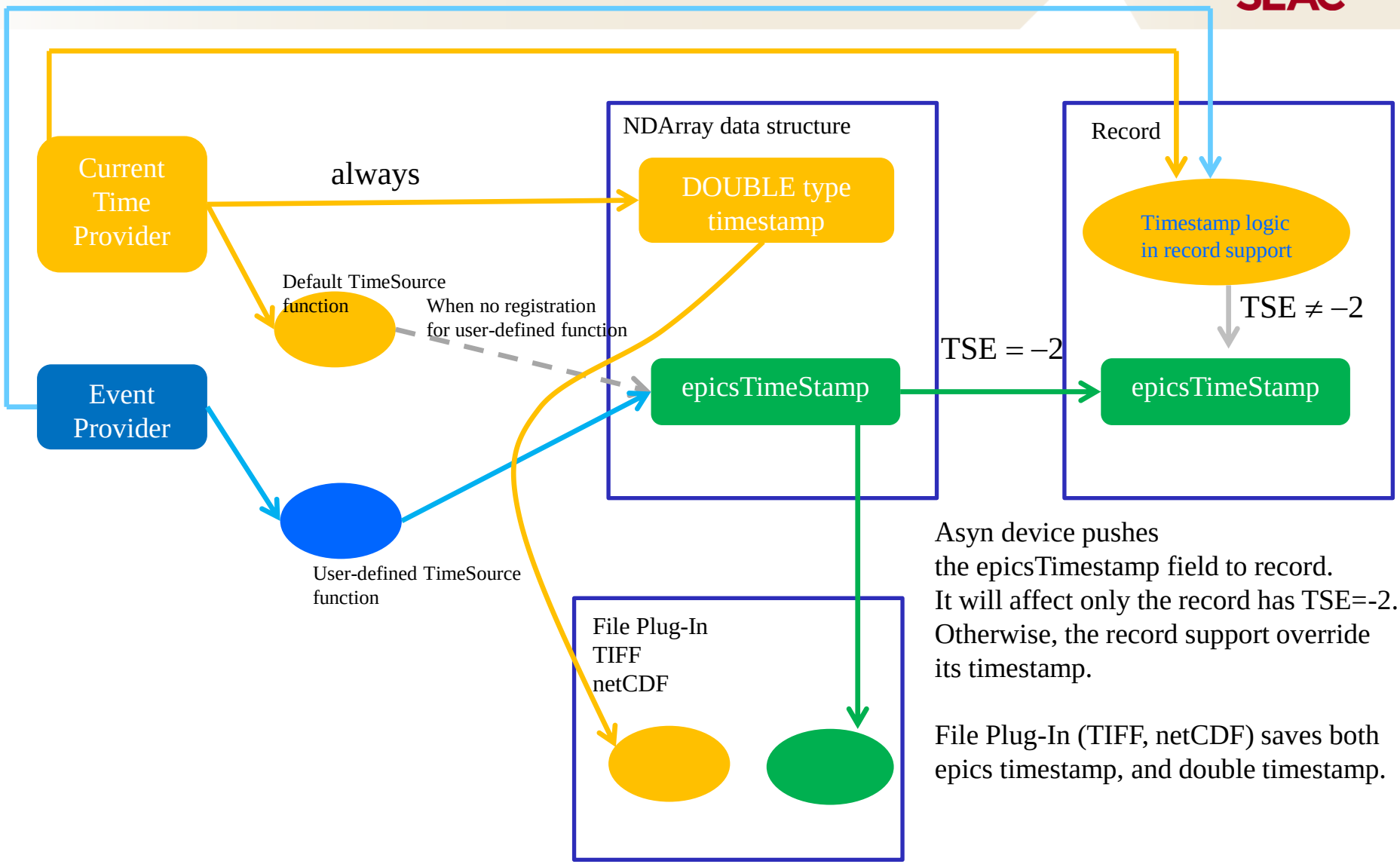
Improvement Details of PV time stamping

SLAC

- Changes on Asyn/areaDector
  - Changes on asynManager and asynPortDriver to support passing timestamps from the driver to device support, and device support to the record.
  - Changes on areaDetector
    - Add up new timestamp field in NDArray data structure
      - Originally, it supports “double” type timestamp NOT epicsTimestamp
      - Now, it has new timestamp field which support the epicsTimestamp
    - The drive sets the new epicsTimestamp field in the NDArray when the image arrives. Then sets all record timestamps for input record in the driver and plugins to be that timestamp from the NDArray.
  - Asyn provides a method to register a user-defined timesstamp callback – TimestampSource, this function will be called by areadDetector right after the image arrives, and updates the epicsTimestamp field in the NDArray
  - If user did not register the timestamp source, there is a default timestamp source which updates the epicsTimestamp field with the Current Time Provider (should be NTP which is chosen by the generaltime facility in EPICS BASE).
  - Note: TSE=-2 is required for record processing to get areaDetector’s “epicsTimestamp field” into an EPICS record’s timestamp.

Time Sources and Time Stamping in the improvements

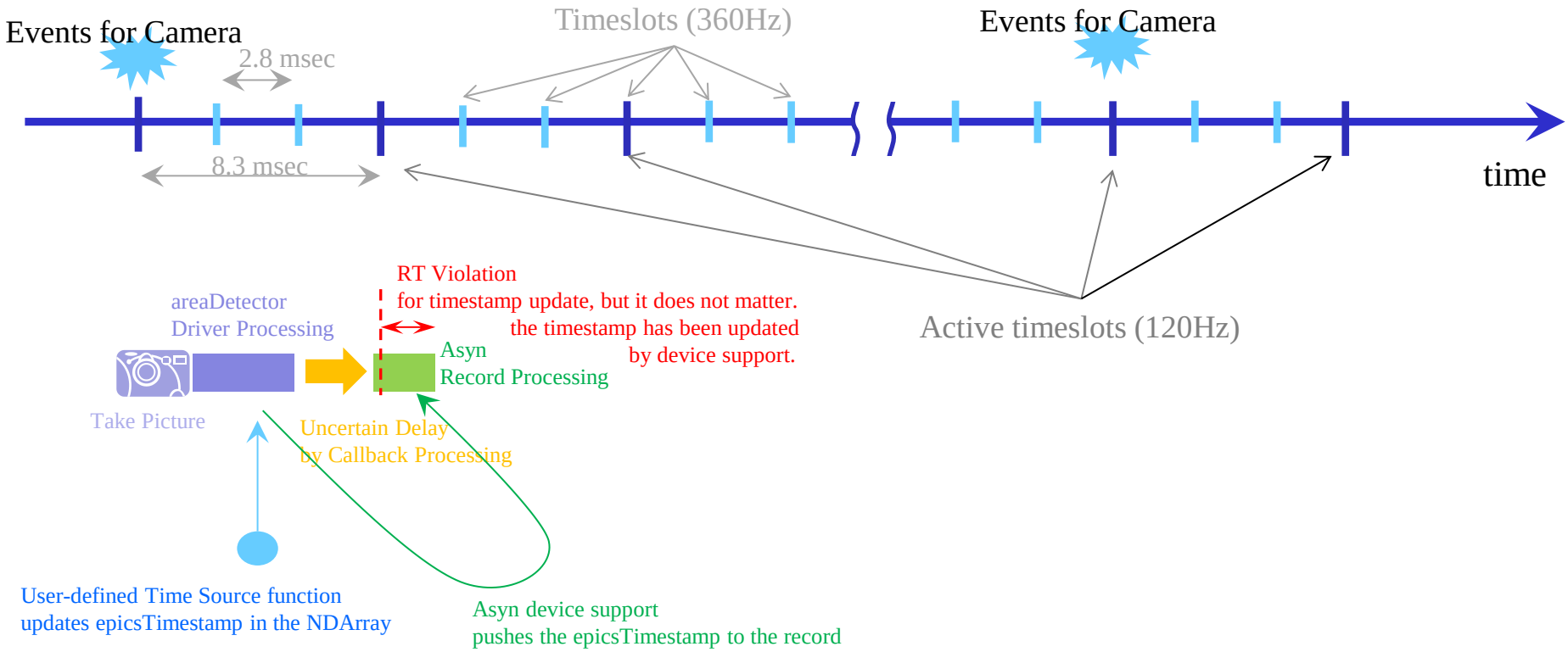
SLAC



Asyn device pushes the epicsTimestamp field to record. It will affect only the record has TSE=-2. Otherwise, the record support override its timestamp.

File Plug-In (TIFF, netCDF) saves both epics timestamp, and double timestamp.

Timeline for the time source callback



Simple Example for using Time Source Callback

SLAC

- Build Application with Mark Rivers' latest areaDetector and Asyn modules
- Set up TSE=-2 for the Image Waveform Record
- Add the following into "st.cmd" to register the "user-defined timestamp source"

```
asynRegisterTimeStampSource("PORT1", "myTimeStampSource")
```

- Prepare "C" code for the user-defined timestamp source

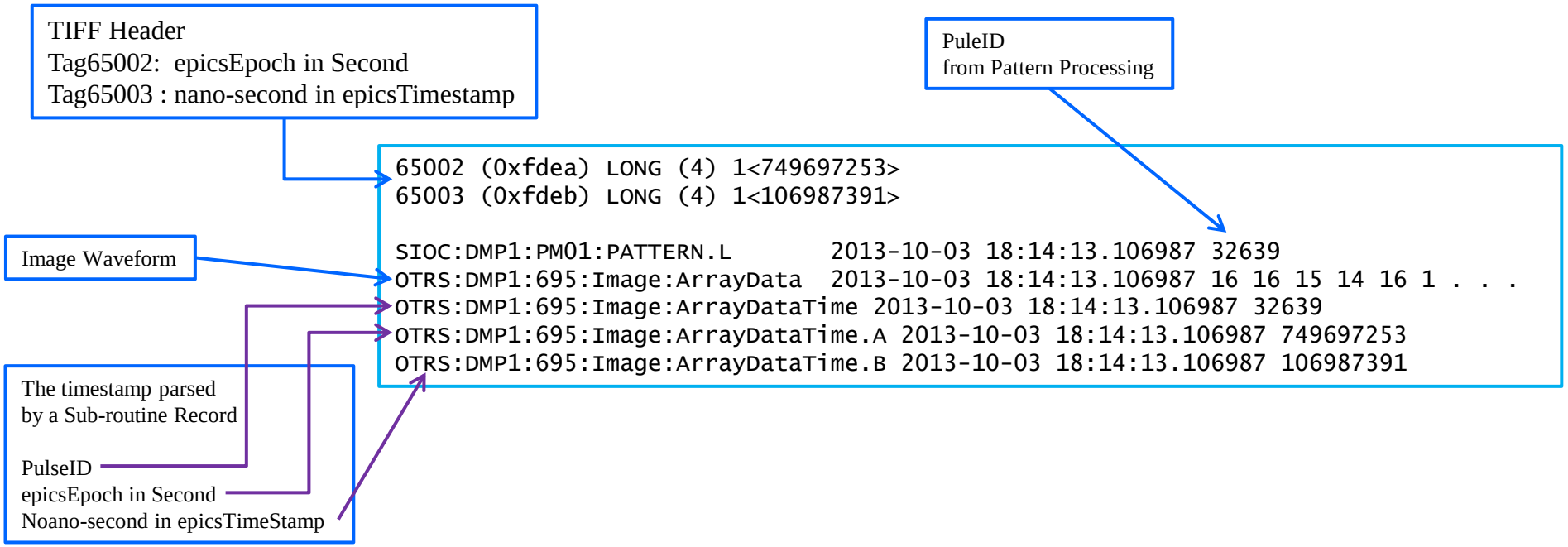
```
#include <epicsTime.h>
#include <registryFunction.h>
#include <epicsExport.h>
#include "evrTime.h"

static void myTimeStampSource(void *userPvt, epicsTimeStamp *pTimeStamp)
{
    evrTimeGet(pTimeStamp, 0)
}
epicsRegisterFunction(myTimeStampSource);
```

In real implementation, the event number is configurable by PV, to bind the timestamp to a specific event

- Add following into DBD file to so we can call this timestamp source function from iocsh()

```
function("myTimeStampSource")
```

Simple Test Example for TIFF file plugin



Attributes for Plugin

SLAC

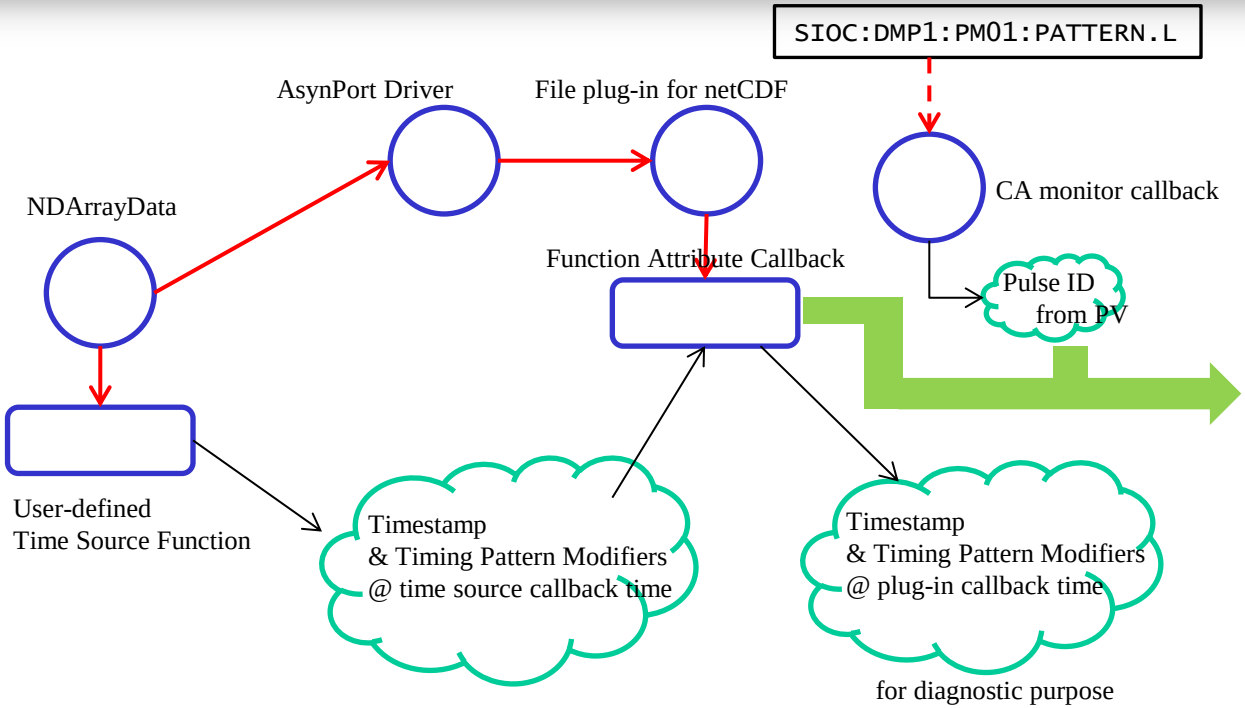
- Plug-In's in areaDetector provides user attributes: EPICS PV attribute and Param attributes
  - To provide additional meta-data which are not supported by the driver and plug-in
  - EPICS PV attribute: get PV value into an attribute
  - Param attribute: turn an internal variable in the software into an attribute
  - User attribute can be configured by use of an XML file
- Implement new attribute called: "function" attribute
  - Provide a callback mechanism to call the user function
  - Allow programming additional logic to existing plug-in's without code change

Test Setup for netCDF file plugin

SLAC

XML file for User Attributes

```
<?xml version="1.0" standalone="no" ?>
<!-- Attributes -->
<Attributes
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://epics.aps.anl.gov/areaDetector/attributes ./attributes.xsd"
    >
    <Attribute name="Seconds"         type="FUNCTION" source="myAttributeFunction"  param="SECONDS"          description="Seconds Past Epoch"/>
    <Attribute name="Nano-Seconds"    type="FUNCTION" source="myAttributeFunction"  param="NANO-SEC"         description="Nano Seconds"/>
    <Attribute name="PulseID"         type="FUNCTION" source="myAttributeFunction"  param="PULSEID"          description="Pulse ID"/>
    <Attribute name="PulseIDPV"       type="EPICS_PV" source="SIOC:DMP1:PM01:PATTERN.L" dbrtype="DBR_NATIVE" description="PulseID via PV"/>
    <Attribute name="TimeSourceTS"    type="FUNCTION" source="myAttributeFunction"  param="TIMESOURCE_TS"    description="Timeslot for timesource"/>
    <Attribute name="Timeslot"        type="FUNCTION" source="myAttributeFunction"  param="TIMESLOT"         description="Timeslot for attribute"/>
    <Attribute name="TimgeString"     type="FUNCTION" source="myAttributeFunction"  param="TIMESTRING"       description="String format time"/>
    <Attribute name="TimeSourceDelay" type="FUNCTION" source="myAttributeFunction"  param="TIMESOURCEDELAY"  description="Time source call delay in nano-sec"/>
    <Attribute name="AttrFuncDelay"   type="FUNCTION" source="myAttributeFunction"  param="ATTRFUNCDELAY"    description="Attr function call delay in nano-sec"/>
</Attributes>
```



netCDF file

Built-in Attributes → epicsTSSec, epicsTSNSec

User Attribute

- Attr_Seconds
- Attr_Nano-Seconds
- Attr_PulseID
- Attr_TimeSourceTS
- Attr_Timeslot
- Attr_TimeString
- Attr_TimeSourceDelay
- Attr_FuncDelay

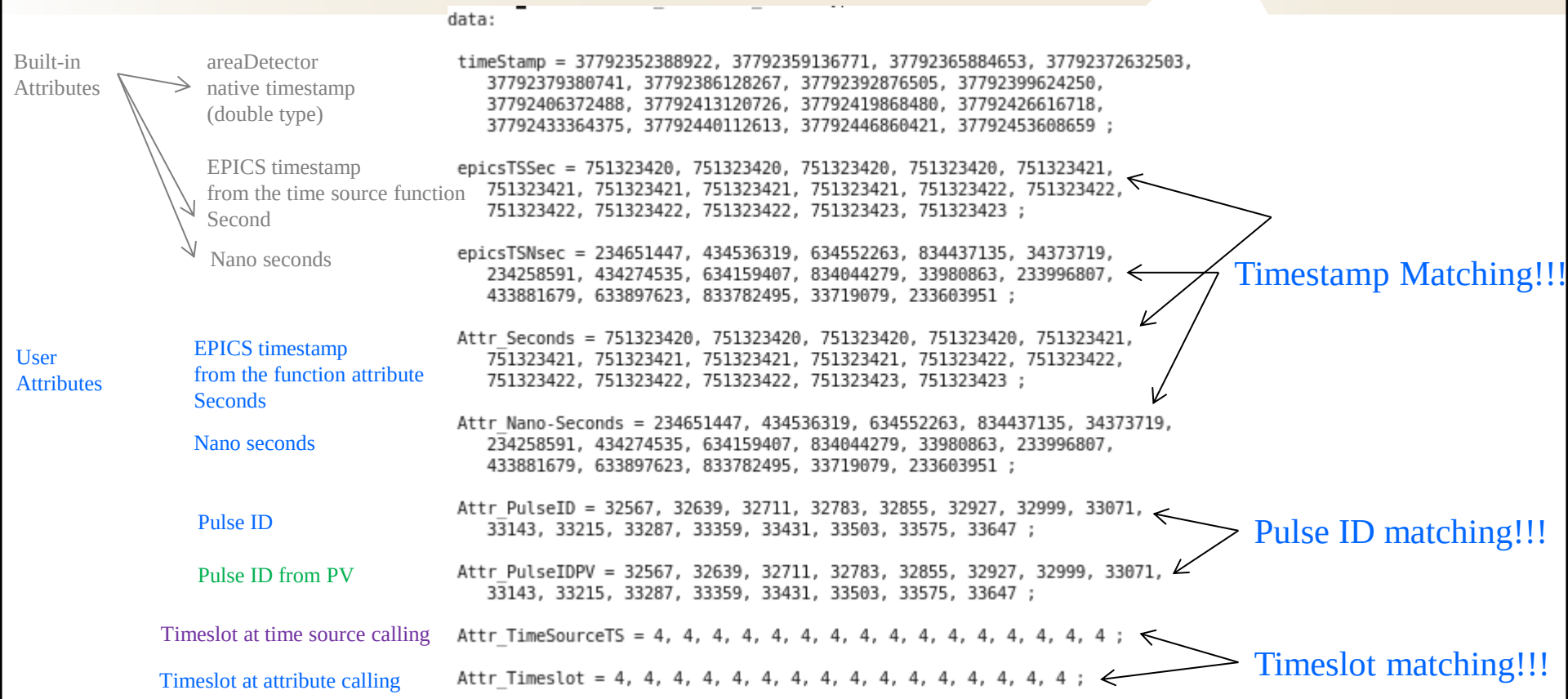
Test Results: dump of netCDF header

SLAC

```
netcdf netCDF_3 {
dimensions:
        numArrays = UNLIMITED ; // (16 currently)
        dim0 = 1038 ;
        dim1 = 1388 ;
        attrStringSize = 256 ;
variables:
        int uniqueId(numArrays) ;
        double timeStamp(numArrays) ;
        int epicsTSSec(numArrays) ;
        int epicsTSNsec(numArrays) ;
        byte array_data(numArrays, dim0, dim1) ;
        int Attr_Seconds(numArrays) ;
        int Attr_Nano-Seconds(numArrays) ;
        int Attr_PulseID(numArrays) ;
        double Attr_PulseIDPV(numArrays) ;
        int Attr_TimeSourceTS(numArrays) ;
        int Attr_Timeslot(numArrays) ;
        char Attr_TimgeString(numArrays, attrString
        float Attr_TimeSourceDelay(numArrays) ;
        float Attr_AttrFuncDelay(numArrays) ;
        int Attr_BayerPattern(numArrays) ;
        int Attr_ColorMode(numArrays) ;

// global attributes:
                :dataType = 1 ;
                :NDNetCDFFileVersion = 3. ;
                :numArrayDims = 2 ;
                :dimSize = 1388, 1038 ;
                :dimOffset = 0, 0 ;
                :dimBinning = 1, 1 ;
                :dimReverse = 0, 0 ;
```

```
:Attr_Seconds_DataType = "UInt32" ;
:Attr_Seconds_Description = "Seconds Past Epoch" ;
:Attr_Seconds_Source = "myAttributeFunction" ;
:Attr_Seconds_SourceType = "NDAttrSourceFunct" ;
:Attr_Nano-Seconds_DataType = "UInt32" ;
:Attr_Nano-Seconds_Description = "Nano Seconds" ;
:Attr_Nano-Seconds_Source = "myAttributeFunction" ;
:Attr_Nano-Seconds_SourceType = "NDAttrSourceFunct" ;
:Attr_PulseID_DataType = "UInt32" ;
:Attr_PulseID_Description = "Pulse ID" ;
:Attr_PulseID_Source = "myAttributeFunction" ;
:Attr_PulseID_SourceType = "NDAttrSourceFunct" ;
:Attr_PulseIDPV_DataType = "Float64" ;
:Attr_PulseIDPV_Description = "PulseID via PV" ;
:Attr_PulseIDPV_Source = "SIOC:DMP1:PM01:PATTERN.L" ;
:Attr_PulseIDPV_SourceType = "NDAttrSourceEPICSPV" ;
:Attr_TimeSourceTS_DataType = "UInt32" ;
:Attr_TimeSourceTS_Description = "Timeslot for timesource" ;
:Attr_TimeSourceTS_Source = "myAttributeFunction" ;
:Attr_TimeSourceTS_SourceType = "NDAttrSourceFunct" ;
:Attr_Timeslot_DataType = "UInt32" ;
:Attr_Timeslot_Description = "Timeslot for attribute" ;
:Attr_Timeslot_Source = "myAttributeFunction" ;
:Attr_Timeslot_SourceType = "NDAttrSourceFunct" ;
:Attr_TimgeString_DataType = "String" ;
:Attr_TimgeString_Description = "String format time" ;
:Attr_TimgeString_Source = "myAttributeFunction" ;
:Attr_TimgeString_SourceType = "NDAttrSourceFunct" ;
:Attr_TimeSourceDelay_DataType = "Float32" ;
:Attr_TimeSourceDelay_Description = "Time source call delay in nano-sec" ;
:Attr_TimeSourceDelay_Source = "myAttributeFunction" ;
:Attr_TimeSourceDelay_SourceType = "NDAttrSourceFunct" ;
:Attr_AttrFuncDelay_DataType = "Float32" ;
:Attr_AttrFuncDelay_Description = "Attr function call delay in nano-sec" ;
:Attr_AttrFuncDelay_Source = "myAttributeFunction" ;
:Attr_AttrFuncDelay_SourceType = "NDAttrSourceFunct" ;
:Attr_BayerPattern_DataType = "Int32" ;
:Attr_BayerPattern_Description = "Bayer Pattern" ;
:Attr_BayerPattern_Source = "Driver" ;
:Attr_BayerPattern_SourceType = "NDAttrSourceDriver" ;
:Attr_ColorMode_DataType = "Int32" ;
:Attr_ColorMode_Description = "Color Mode" ;
:Attr_ColorMode_Source = "Driver" ;
:Attr_ColorMode_SourceType = "NDAttrSourceDriver" ;
```

SLAC


data:
Built-in Attributes
areaDetector native timestamp (double type)
timeStamp = 37792352388922, 37792359136771, 37792365884653, 37792372632503, 37792379380741, 37792386128267, 37792392876505, 37792399624250, 37792406372488, 37792413120726, 37792419868480, 37792426616718, 37792433364375, 37792440112613, 37792446860421, 37792453608659 ;
EPICS timestamp from the time source function Second
epicsTSSec = 751323420, 751323420, 751323420, 751323420, 751323421, 751323421, 751323421, 751323421, 751323421, 751323422, 751323422, 751323422, 751323422, 751323422, 751323423, 751323423 ;
Nano seconds
epicsTSNsec = 234651447, 434536319, 634552263, 834437135, 34373719, 234258591, 434274535, 634159407, 834044279, 33980863, 233996807, 433881679, 633897623, 833782495, 33719079, 233603951 ;
Timestamp Matching!!!
User Attributes
EPICS timestamp from the function attribute Seconds
Attr_Seconds = 751323420, 751323420, 751323420, 751323420, 751323421, 751323421, 751323421, 751323421, 751323421, 751323422, 751323422, 751323422, 751323422, 751323422, 751323423, 751323423 ;
Nano seconds
Attr_Nano-Seconds = 234651447, 434536319, 634552263, 834437135, 34373719, 234258591, 434274535, 634159407, 834044279, 33980863, 233996807, 433881679, 633897623, 833782495, 33719079, 233603951 ;
Pulse ID
Attr_PulseID = 32567, 32639, 32711, 32783, 32855, 32927, 32999, 33071, 33143, 33215, 33287, 33359, 33431, 33503, 33575, 33647 ;
Pulse ID matching!!!
Pulse ID from PV
Attr_PulseIDPV = 32567, 32639, 32711, 32783, 32855, 32927, 32999, 33071, 33143, 33215, 33287, 33359, 33431, 33503, 33575, 33647 ;
Timeslot at time source calling
Attr_TimeSourceTS = 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4 ;
Timeslot matching!!!
Timeslot at attribute calling
Attr_Timeslot = 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4 ;

String Type
EPICS timestamp

```
Attr_TimgeString =
 "2013-10-22 13:57:00.234651",
 "2013-10-22 13:57:00.434536",
 "2013-10-22 13:57:00.634552",
 "2013-10-22 13:57:00.834437",
 "2013-10-22 13:57:01.034374",
 "2013-10-22 13:57:01.234259",
 "2013-10-22 13:57:01.434275",
 "2013-10-22 13:57:01.634159",
 "2013-10-22 13:57:01.834044",
 "2013-10-22 13:57:02.033981",
 "2013-10-22 13:57:02.233997",
 "2013-10-22 13:57:02.433882",
 "2013-10-22 13:57:02.633898",
 "2013-10-22 13:57:02.833782",
 "2013-10-22 13:57:03.033719",
 "2013-10-22 13:57:03.233604" ;
```

Delay measurement
for the time source callback
from the fiducial
(in nano-second scale)

```
Attr_TimeSourceDelay = 2452740, 2448975, 2448050, 2443815, 2450084, 2428840,
   2440866, 2435302, 2443899, 2451916, 2441580, 2441395, 2432193, 2448941,
   2441656, 2449101 ;
```

Delay measurement
for the attribute callback
from the fiducial
(in nano-second scale)

```
Attr_AttrFuncDelay = 2494605, 2485008, 2484017, 2485319, 2485521, 2465958,
   2475521, 2469034, 2478344, 2489866, 2478899, 2476109, 2468748, 2507740,
   2474874, 2487479 ;
}
```

Only few tens micro-second delay
between the user-defined time source callback
and the attribute callback

Conclusion

SLAC

- areaDetector improvements
  - time source callback
  - user attributes in plugin
- Excellent co-operation from Mark River
  - Accept to apply SLAC specific requirements to areaDetector
  - Shift the specific requirements to generic new features for benefit to community

SLAC

Special Thanks to Mark Rivers !!!