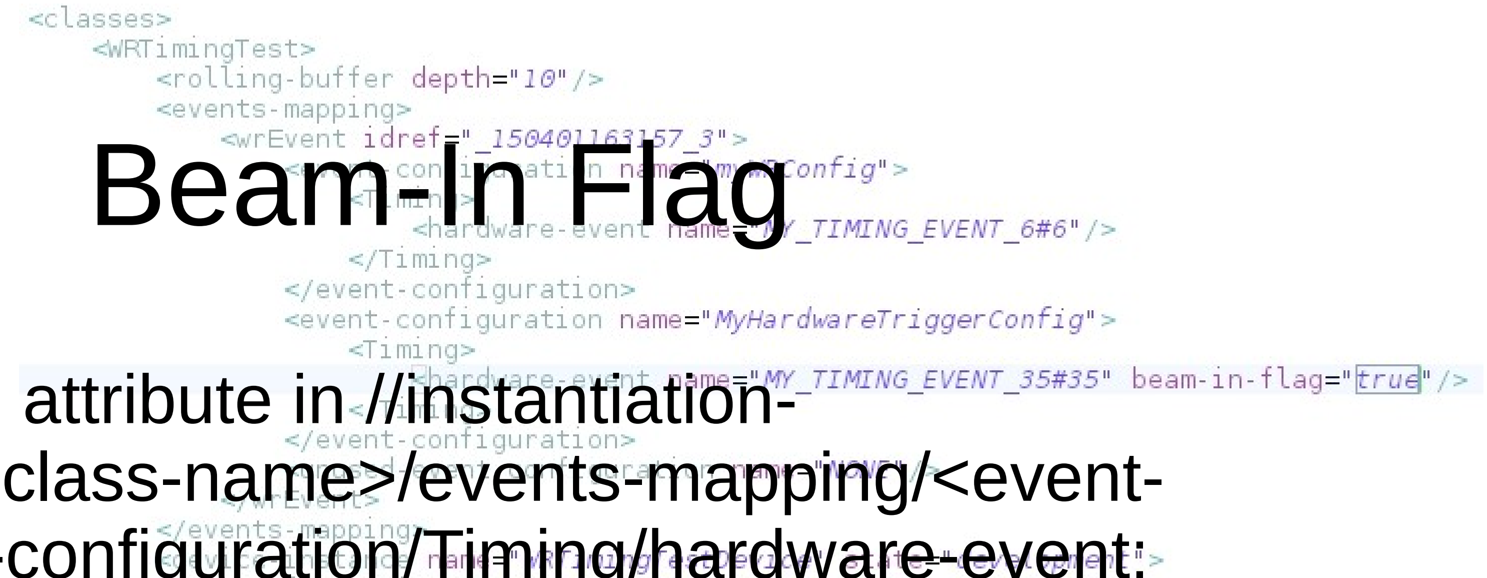
FESA3 5.0.1
„Sneak Preview“

What‘s new @GSI?

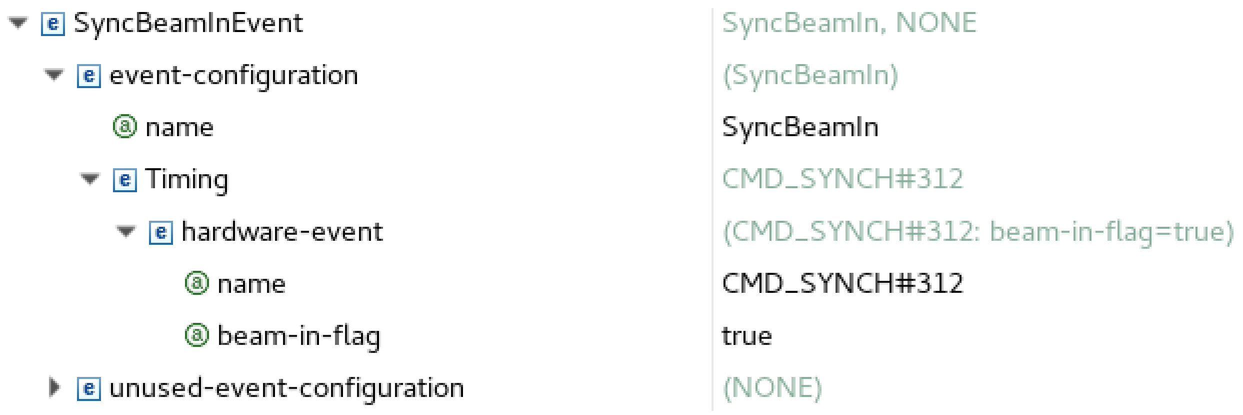
- Event triggering on beam-in flag
- Offset timing events
- Multiple timing receivers
- Multiplexing per beam production chain
- Enabling process statistics via CMX
- C++11 enabled throughout

Beam-In Flag



```
<classes>
    <WRTimingTest>
        <rolling-buffer depth="10"/>
        <events-mapping>
            <wrEvent idref="_150401163157_3">
                <event-configuration name="myWRConfig">
                    <Timing>
                        <hardware-event name="MY_TIMING_EVENT_6#6"/>
                    </Timing>
                </event-configuration>
                <event-configuration name="MyHardwareTriggerConfig">
                    <Timing>
                        <hardware-event name="MY_TIMING_EVENT_35#35" beam-in-flag="true"/>
                    </Timing>
                </event-configuration>
                <unused-event-configuration name="NONE"/>
            </wrEvent>
        </events-mapping>
        <device-instance name="WRTimingTestDevice" state="development">
```

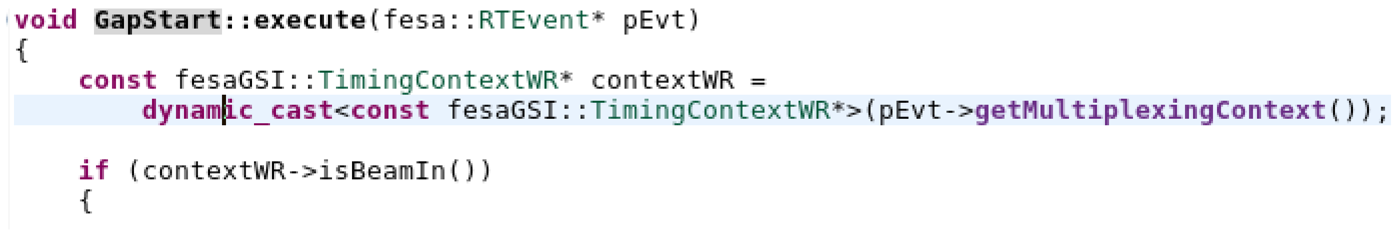
- **New**: optional attribute in //instantiation-unit/classes/<class-name>/events-mapping/<event-name>/event-configuration/Timing/hardware-event: @beam-in .
- If an action is required for only one BeamIn state, use the Event Mapping
- BeamInFlag attribute in hardware-event
- Timing hardware will only trigger on the desired flag state



SyncBeamInEvent	SyncBeamIn, NONE
event-configuration	(SyncBeamIn)
name	SyncBeamIn
Timing	CMD_SYNCH#312
hardware-event	(CMD_SYNCH#312: beam-in-flag=true)
name	CMD_SYNCH#312
beam-in-flag	true
unused-event-configuration	(NONE)

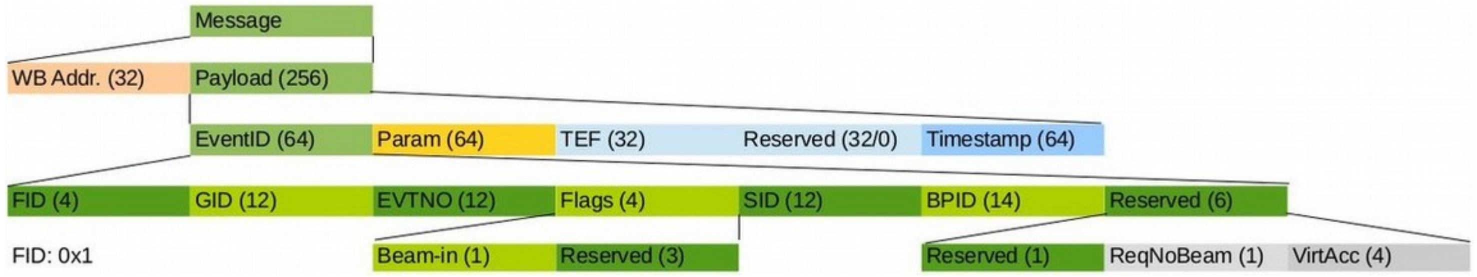
BeamIn Flag in RTAction

- If a different action is required for BeamIn and BeamOut, use a single RTAction and check the context for isBeamIn



```
void GapStart::execute(fesa::RTEvent* pEvt)
{
    const fesaGSI::TimingContextWR* contextWR =
        dynamic_cast<const fesaGSI::TimingContextWR*>(pEvt->getMultiplexingContext());

    if (contextWR->isBeamIn())
    {
```

Event Triggering



- The timing receiver can only match a contiguous block from the MSB
- Normal FESA event matches FID/Group/EventNo
- Matching e.g. EventNo/BP is not possible
- Creating two events BeamIn/Out uses more timing receiver resources.

Creating Timing Events

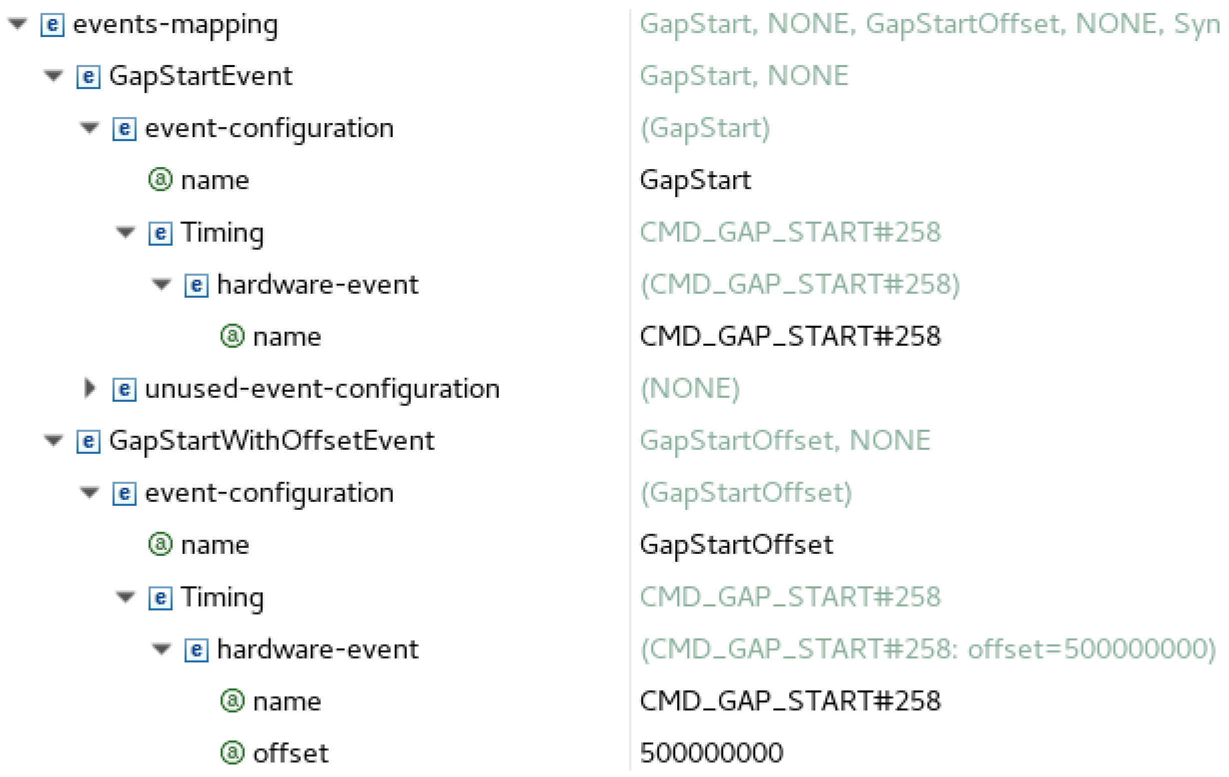
- /opt/fesa/nfs/global/scripts/inject-event-id.sh
- BeamIn Event:
  - inject-event-id.sh 210 312 1 1 1 8
- BeamOut Event:
  - inject-event-id.sh 210 312 1 1 1 0

Offset Timing Events

- The timing receiver can be programmed to generate an RTAction up to 1s after or 0.1s before the event
- An offset event behaves as a normal event
- Uses the same timing receiver resources
- If they are too close some will be marked as delayed

Event Configuration

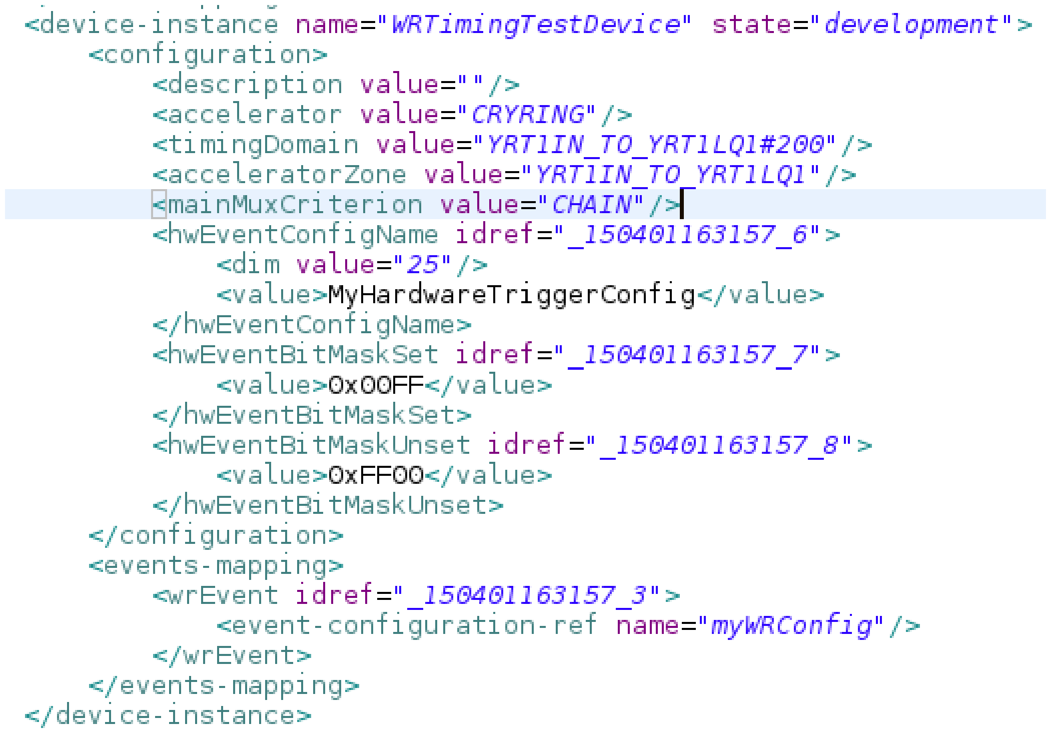
- Optional offset attribute in event-configuration



events-mapping	GapStart, NONE, GapStartOffset, NONE, Syn
GapStartEvent	GapStart, NONE
event-configuration	(GapStart)
name	GapStart
Timing	CMD_GAP_START#258
hardware-event	(CMD_GAP_START#258)
name	CMD_GAP_START#258
unused-event-configuration	(NONE)
GapStartWithOffsetEvent	GapStartOffset, NONE
event-configuration	(GapStartOffset)
name	GapStartOffset
Timing	CMD_GAP_START#258
hardware-event	(CMD_GAP_START#258: offset=500000000)
name	CMD_GAP_START#258
offset	500000000

Multiplexing per Beam Production Chain

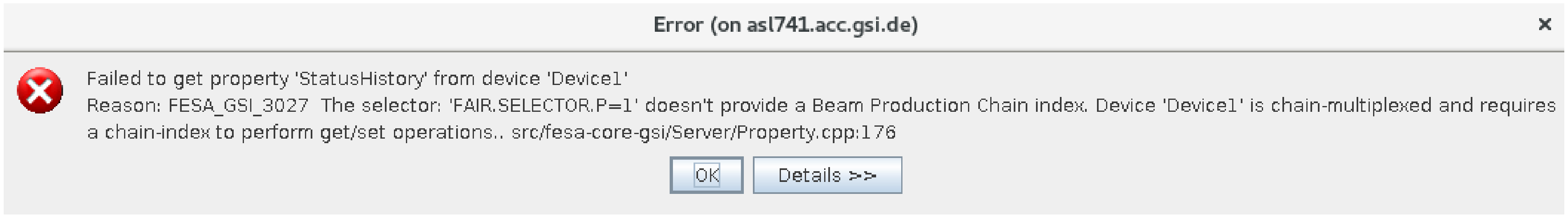
- Set mainMuxCriterion for the device in the instance file



```
<device-instance name="WRTimingTestDevice" state="development">
    <configuration>
        <description value=""/>
        <accelerator value="CRYRING"/>
        <timingDomain value="YRT1IN_TO_YRT1LQ1#200"/>
        <acceleratorZone value="YRT1IN_TO_YRT1LQ1"/>
        <mainMuxCriterion value="CHAIN"/>
        <hwEventConfigName idref="_150401163157_6">
            <dim value="25"/>
            <value>MyHardwareTriggerConfig</value>
        </hwEventConfigName>
        <hwEventBitMaskSet idref="_150401163157_7">
            <value>0x00FF</value>
        </hwEventBitMaskSet>
        <hwEventBitMaskUnset idref="_150401163157_8">
            <value>0xFF00</value>
        </hwEventBitMaskUnset>
    </configuration>
    <events-mapping>
        <wrEvent idref="_150401163157_3">
            <event-configuration-ref name="myWRConfig"/>
        </wrEvent>
    </events-mapping>
</device-instance>
```

Multiplexing per Beam Production Chain

- Behaves the same as sequence or BP multiplexing



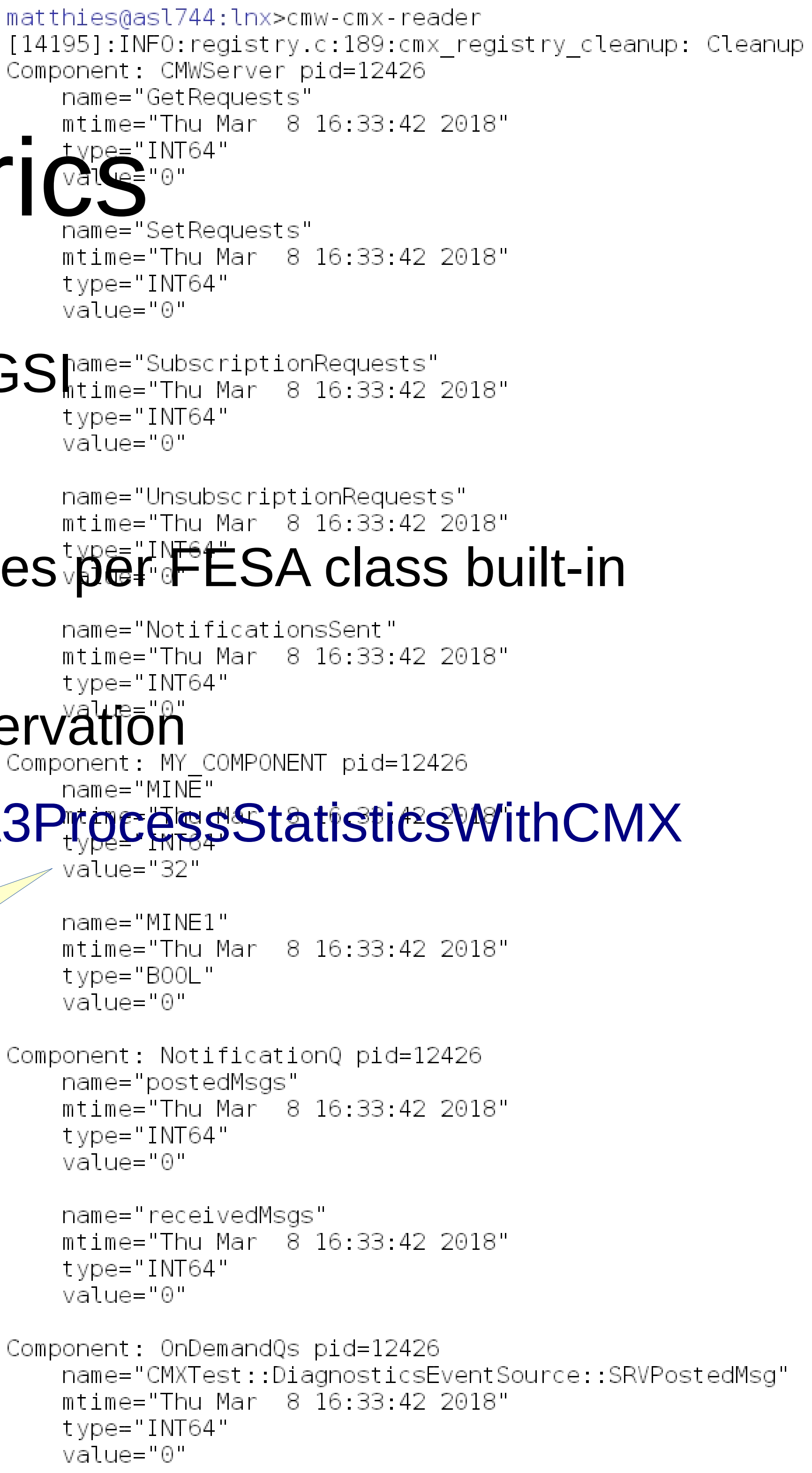
```
Failed to get property 'StatusHistory' from device 'Device1'
Reason: FESA_GSI_3027  The selector: 'FAIR.SELECTOR.P=1' doesn't provide a Beam Production Chain index. Device 'Device1' is chain-multiplexed and requires
a chain-index to perform get/set operations.. src/fesa-core-gsi/Server/Property.cpp:176
```

- Get/Set work with FAIR.SELECTOR.C=x
- Timing implementation difference: The ChainID is in the Timing Payload, not the EventID.
- Event matching on chainID will never be possible

CMX Metrics

- Enabling process statistics in FESA3@GSI
- Support for diagnostics and debugging
- Default set of potentially interesting values per FESA class built-in (notification queue length etc.)
- Possibility to add custom values for observation
- https://www-acc.gsi.de/wiki/FESA/FESA3ProcessStatisticsWithCMX

```
#include <fesa-core/Diagnostic/MetricsManager.h>

static int32_t integer = 32;
static bool boolean = false;
metricsManager.registerMetric(„MY_COMPONENT‟, „MINE‟, integer);
metricsManager.registerMetric(„MY_COMPONENT‟, „MINE1‟, boolean);
```

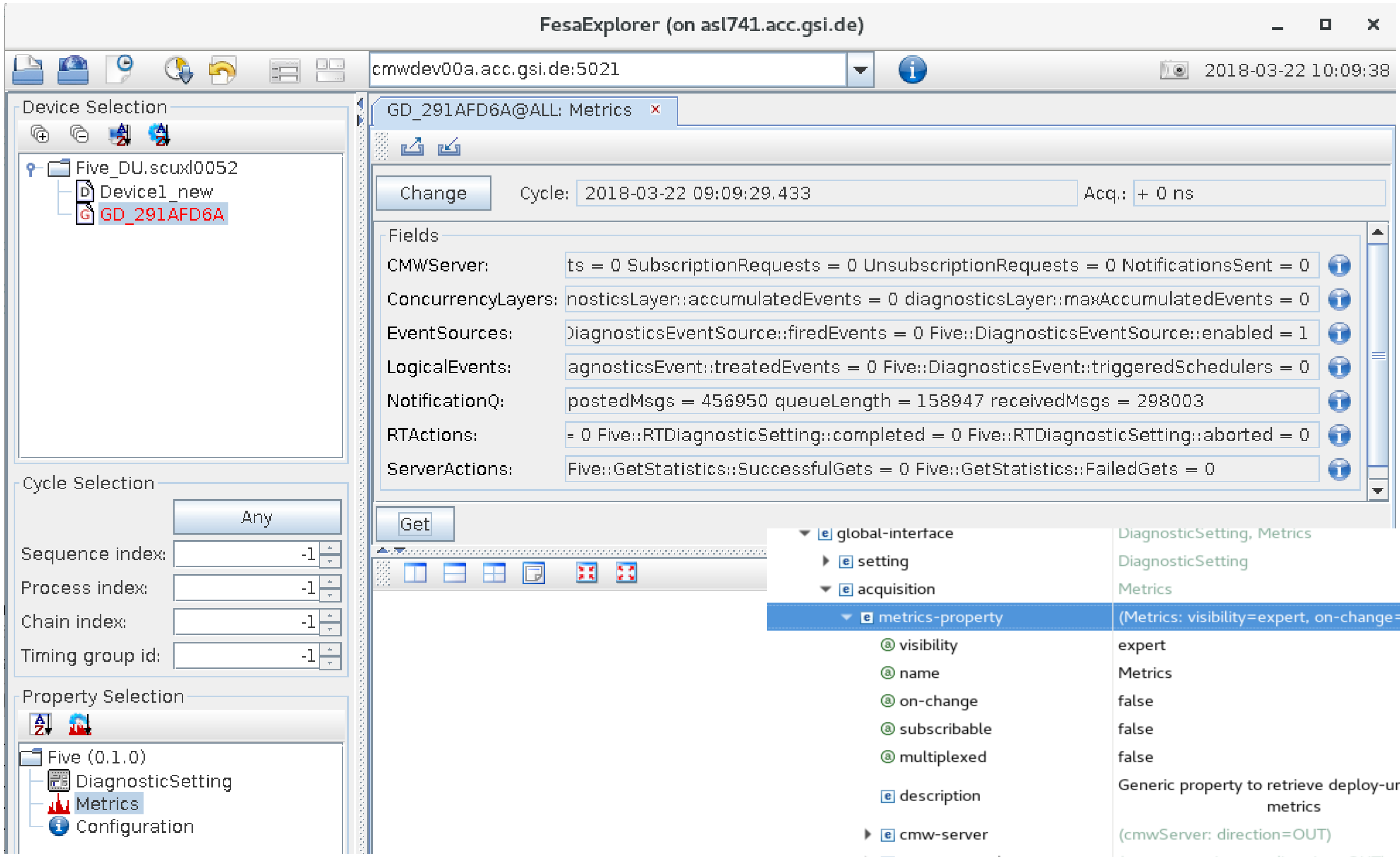


```
matthies@asl744:lnx>cmw-cmx-reader
[14195]:INFO:registry.c:189:cmx_registry_cleanup: Cleanup
Component: CMWServer pid=12426
    name="GetRequests"
    mtime="Thu Mar  8 16:33:42 2018"
    type="INT64"
    value="0"

    name="SetRequests"
    mtime="Thu Mar  8 16:33:42 2018"
    type="INT64"
    value="0"

    name="SubscriptionRequests"
    mtime="Thu Mar  8 16:33:42 2018"
    type="INT64"
    value="0"

    name="UnsubscriptionRequests"
    mtime="Thu Mar  8 16:33:42 2018"
    type="INT64"
    value="0"

    name="NotificationsSent"
    mtime="Thu Mar  8 16:33:42 2018"
    type="INT64"
    value="0"
Component: MY_COMPONENT pid=12426
    name="MINE"
    mtime="Thu Mar  8 16:33:42 2018"
    type="INT64"
    value="32"

    name="MINE1"
    mtime="Thu Mar  8 16:33:42 2018"
    type="BOOL"
    value="0"

Component: NotificationQ pid=12426
    name="postedMsgs"
    mtime="Thu Mar  8 16:33:42 2018"
    type="INT64"
    value="0"

    name="receivedMsgs"
    mtime="Thu Mar  8 16:33:42 2018"
    type="INT64"
    value="0"

Component: OnDemandQs pid=12426
    name="CMXTest::DiagnosticsEventSource::SRVPostedMsg"
    mtime="Thu Mar  8 16:33:42 2018"
    type="INT64"
    value="0"
```

Notification Queues

- Notification queues overflowing has caused errors
- Notification runs at lower priority
- Example class:
  - RT Action at 1 kHz issues 20 manual notifications
- 10,000 Notifications/s is the practical limit for an SCU
- New: CMX metric entry for notification queue length

Notification Queue CMX Metrics



FesaExplorer (on asl741.acc.gsi.de)

cmwdev00a.acc.gsi.de:5021 2018-03-22 10:09:38

Device Selection

- Five_DU.scuxl0052
  - Device1_new
  - GD_291AFD6A

Cycle Selection

	Any
Sequence index:	-1
Process index:	-1
Chain index:	-1
Timing group id:	-1

Property Selection

- Five (0.1.0)
  - DiagnosticSetting
  - Metrics
  - Configuration

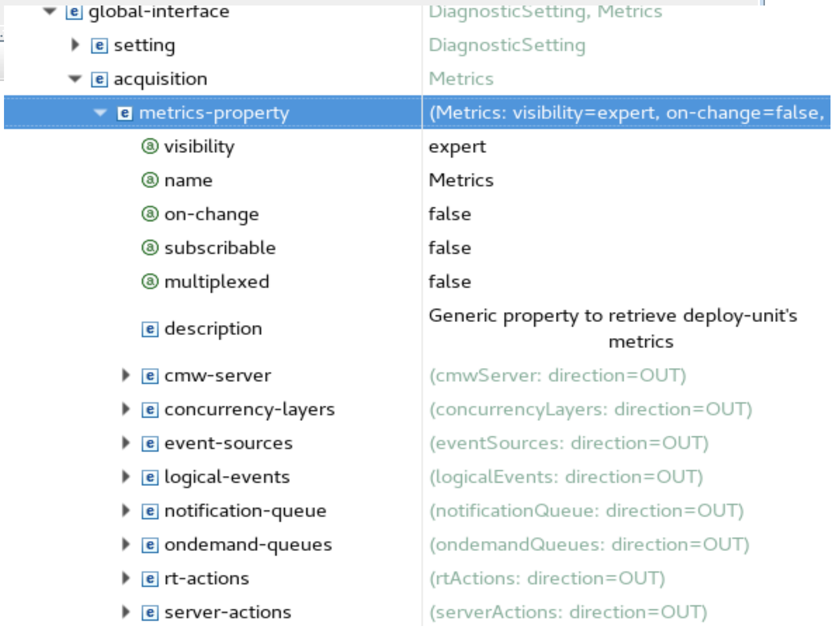
GD_291AFD6A@ALL: Metrics

Change Cycle: 2018-03-22 09:09:29.433 Acq.: + 0 ns

Fields

CMWServer:	ts = 0 SubscriptionRequests = 0 UnsubscriptionRequests = 0 NotificationsSent = 0
ConcurrencyLayers:	nosticsLayer::accumulatedEvents = 0 diagnosticsLayer::maxAccumulatedEvents = 0
EventSources:	)iagnosticsEventSource::firedEvents = 0 Five::DiagnosticsEventSource::enabled = 1
LogicalEvents:	agnosticsEvent::treatedEvents = 0 Five::DiagnosticsEvent::triggeredSchedulers = 0
NotificationQ:	postedMsgs = 456950 queueLength = 158947 receivedMsgs = 298003
RTActions:	= 0 Five::RTDiagnosticSetting::completed = 0 Five::RTDiagnosticSetting::aborted = 0
ServerActions:	Five::GetStatistics::SuccessfulGets = 0 Five::GetStatistics::FailedGets = 0

Get



global-interface	DiagnosticSetting, Metrics
setting	DiagnosticSetting
acquisition	Metrics
metrics-property	(Metrics: visibility=expert, on-change=false,
visibility	expert
name	Metrics
on-change	false
subscribable	false
multiplexed	false
description	Generic property to retrieve deploy-unit's metrics
cmw-server	(cmwServer: direction=OUT)
concurrency-layers	(concurrencyLayers: direction=OUT)
event-sources	(eventSources: direction=OUT)
logical-events	(logicalEvents: direction=OUT)
notification-queue	(notificationQueue: direction=OUT)
ondemand-queues	(ondemandQueues: direction=OUT)
rt-actions	(rtActions: direction=OUT)
server-actions	(serverActions: direction=OUT)

Improvements

- Subscriptions: adaption of first update behaviour to avoid exceptions
  - Makes subscription to ALL usable
- Removal of implementation for „old“ timing selector format
  - FAIR.BP.x

Bug-Fixes

- Subset definition uses vardim element instead of vardim1
- Timing event Nos SEQ_START, BP_START work with offsets
- Timing events for devices in different groups (also 4.3.1 patch)
- Timing event performance improvements

Backward incompatible Changes

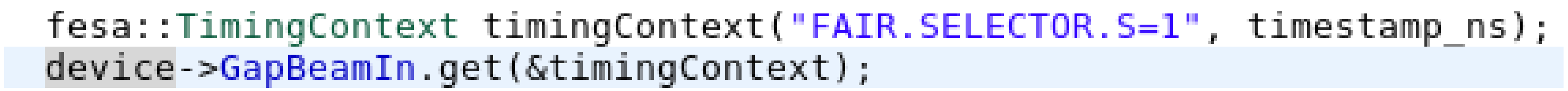
- DataStore::getName() returns a **const** string reference instead of a string.
- The enumeration value MultiplexingContext::TimerCtxt has been removed because it was never used
- Generated struct types now have a default constructor that initialize all their attributes to valid values (0 or, for enumerations without a value that maps to 0, the first value of the enumeration)
  - → struct types cannot be initialized with a C-style initializer anymore

Timing Configuration Backward Incompatible Changes

- Zero is now a valid Beam Process/Sequence Index.
- In previous versions zero in the selector would match ALL

„No NoneContext“

- It is not possible anymore to get and set a value of a multiplexed acquisition field with a NoneContext. This was allowed in the past FESA FWK versions, but was a bug. Trying to do so will now throw an exception.
  - Consider if the field should be multiplexed
  - Provide your own acquisition context with a selector and local timestamp



```
fesa::TimingContext timingContext("FAIR.SELECTOR.S=1", timestamp_ns);
device->GapBeamIn.get(&timingContext);
```

- Future feature: get anything – retrieve an entry from the acquisition buffer – similar to subscription to ALL

Release Notes

- GSI:

  https://www-acc.gsi.de/wiki/FESA/FESA3ReleaseHistory
- CERN: https://www-acc.gsi.de/wiki/FESA/FESA3500#relnotescern