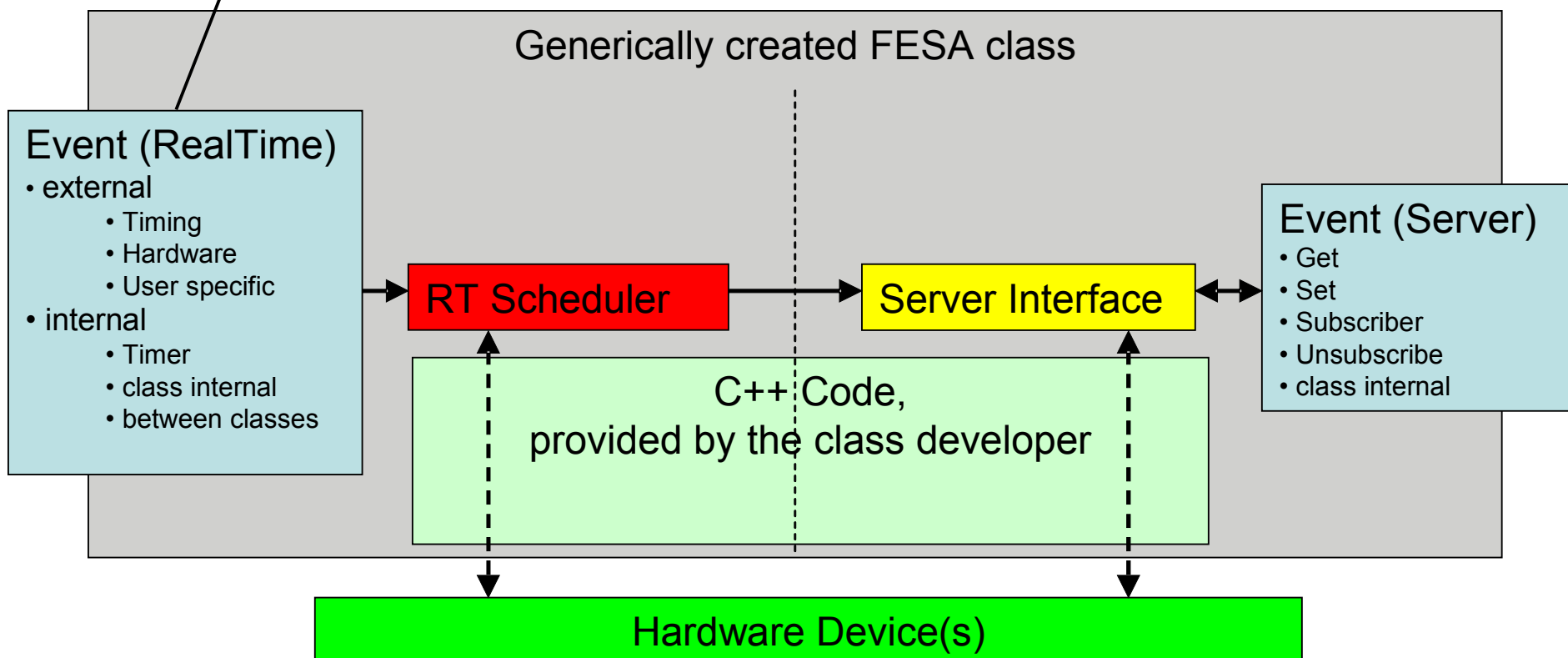




Event Sources

source	cardinality	class specific	payload possible
• TimingEventSource	(0..1)	no	yes
• TimerEventSource	(0..1)	no	no
• OnSubscriptionEventSource	(0..1)	no	yes
• OnDemandEventSource	(0..n)	yes	yes
• CustomEventSource	(0..n)	yes	yes





enable/disable events and sources



```
if (data.itemAvailable("VA_EnableLongTimerEvent"))
{
    bool enableLongTimerEvent = data.VA_EnableLongTimerEvent.get();
    logTextStream << "\nset event long timer enable=" << enableLongTimerEvent;
    if (enableLongTimerEvent)
    {
        this->Ex04ServiceLocator->enableRTEEvent("LE_NeedsLongTermCalibration", "Timer");
    }
    else
    {
        this->Ex04ServiceLocator->disableRTEEvent("LE_NeedsLongTermCalibration", "Timer");
    }
}
```

name of the logical-event

service-locator of class Ex04

deprecated argument, can be ignored

```
if (data.itemAvailable("VA_EnableOnDemandEventSource"))
{
    bool enableOnDemandEventSource = data.VA_EnableOnDemandEventSource.get();
    logTextStream << "\nset onDemand event source enable=" << enableOnDemandEventSource;
    if (enableOnDemandEventSource)
    {
        this->Ex04ServiceLocator->enableEventSource("ManualCalibrationEventSource");
    }
    else
    {
        this->Ex04ServiceLocator->disableEventSource("ManualCalibrationEventSource");
    }
}
```

name of the event-source



TimerEventSource

- any number of software-timers
- endless-loop for each timer
- fixed name
- currently always present in class-design (to be fixed)
- no payload
- Period in [ms]

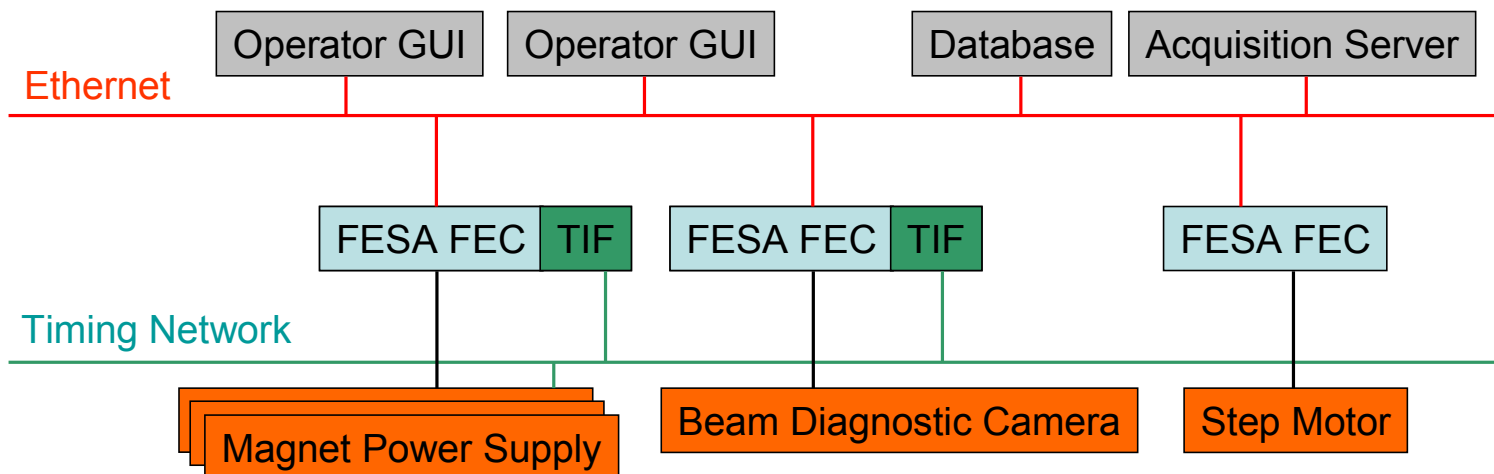


▼ [e] events
▼ [e] sources
▼ [e] timer-event-source
@a name
Timer

▼ [e] event-configuration	(Timing Timer O
@a name	MyTimerConfig
▼ [e] Timer	(timer-event+)
▼ [e] timer-event	
@a period	1500
▼ [e] timer-event	
@a period	3000



TimingEventSource



- any number of different timing-events
- fixed name
- currently always present in class-design (to be fixed)
- payload

Machine	EventCode	Sequence	Beam Proc.	Prod.Chain	Exec.Stamp	?
---------	-----------	----------	------------	------------	------------	---

<ul style="list-style-type: none"> events <ul style="list-style-type: none"> sources <ul style="list-style-type: none"> timing-event-source <ul style="list-style-type: none"> name 	<ul style="list-style-type: none"> event-configuration <ul style="list-style-type: none"> name: MyTimingConfig Timing <ul style="list-style-type: none"> hardware-event <ul style="list-style-type: none"> name: START_CYCLE#CTIM#32 hardware-event <ul style="list-style-type: none"> name: SRC_DST_ID#CTIM#31
--	--



OnDemandEventSource



- **ServerAction** triggers **RTAction**
- **RTAction** triggers **RTAction**
- any number of class-specific instances
- manual or automatic trigger
- payload

▼ [e] events	(sources?, logical-events?)
▼ [e] sources	(timing-event-source, tim
▼ [e] on-demand-event-source	(description*)
@a name	MyOnDemandSource1
▼ [e] on-demand-event-source	(description*)
@a name	MyOnDemandSource2

```
AbstractAction::triggerOnDemandEventSource("MyOnDemandEventSource",context,0);
```

```
std::string payload = "myPayload";
```

```
AbstractAction::triggerOnDemandEventSource("MyOnDemandEventSource",context,payload.c_str(),payload.size(),0);
```

▼ [e] rt-action	((description
@a name	MyAction
▼ [e] triggered-event-source	
@a on-demand-event-source-ref	MyOnDemandSource1
@a automatic	true
▶ [e] triggered-event-source	

▼ [e] event-configuration	(Timing Timer OnDer
@a name	MyOnDemandConfig
▼ [e] OnDemand	(on-demand-event-sour
▼ [e] on-demand-event-source-ref	
@a name	MyOnDemandSource1
▼ [e] on-demand-event-source-ref	
@a name	MyOnDemandSource2



OnSubscriptionEventSource



- subscribe to properties of other classes
- get triggered whenever there is new data
- payload = received data
- (not yet fully working at GSI)

<ul style="list-style-type: none"> ▼ [e] events <ul style="list-style-type: none"> ▼ [e] sources <ul style="list-style-type: none"> ▼ [e] on-subscription-event-source <ul style="list-style-type: none"> ⓐ name 	<ul style="list-style-type: none"> ▼ [e] relationship <ul style="list-style-type: none"> ▼ [e] association <ul style="list-style-type: none"> [e] class-name 	<ul style="list-style-type: none"> (association*, composition*, (class-name, class-major-ve FFTONSubscriptionSrc (description?) OnSubscription
---	---	---

<ul style="list-style-type: none"> ▼ [e] event-configuration <ul style="list-style-type: none"> ⓐ name ▼ [e] OnSubscription <ul style="list-style-type: none"> ▼ [e] on-subscription-event <ul style="list-style-type: none"> ⓐ context ⓐ device ⓐ property ▼ [e] on-subscription-event <ul style="list-style-type: none"> ⓐ context ⓐ device ⓐ property 	<ul style="list-style-type: none"> (Timing Timer OnDemand C MyOnSubscriptionConfig (on-subscription-event+) SIS.USER.VACC_12 MyDevice1 PropertyTriggerOnSubscription NONE MyDevice2 PropertyTriggerOnSubscription
---	--



CustomEventSource



- write your own event-source
- any number of class-specific custom-event-sources
- any number of custom-events for each custom-event-source
- payload

▼ [e] event-configuration	(Timing Timer OnF
@a name	MultiCustomConfig
▼ [e] Custom	(MyCustomSource*)
▼ [e] MyCustomSource	
@a custom-event	customEvent1
▼ [e] MyCustomSource	
@a custom-event	customEvent2

▼ [e] events	
▼ [e] sources	
▼ [e] custom-event-source	(description+, cust
@a name	MyCustomSource
▼ [e] custom-event	
@a name	customEvent1
▼ [e] custom-event	
@a name	customEvent2



CustomEventSource



```
void MyCustomSource::wait(boost::shared_ptr<fesa::RTEvent>& eventToFire)
{
    // This code provides an example on how a custom-event-source can look like.
    struct timespec time;
    time.tv_sec = 1;    // wake-up every 1.5 seconds
    time.tv_nsec = 500000000;

    // Typically there is some blocking call, like this one in a event-source.
    nanosleep(&time, &time);

    if(condition)
    {
        createEvent(eventToFire, MyCustomSource::customEvent1);
    }
    else
    {
        createEvent(eventToFire, MyCustomSource::customEvent2);
    }
}
```

some blocking call

fesa-core