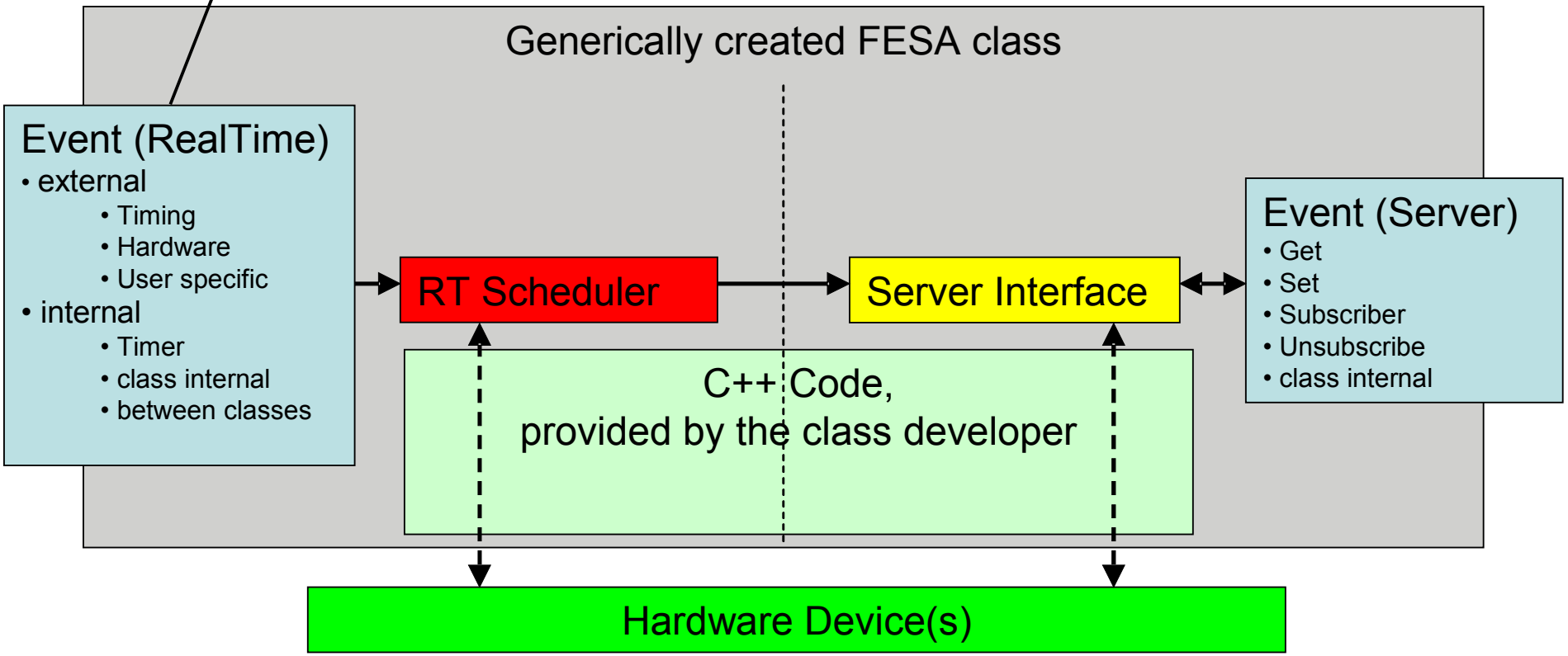
Event Sources

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_E
    logTextStream << "\nset event long timer
    if (enableLongTimerEvent)
    {
        this->Ex04ServiceLocator_->enableRTEvent("LE_NeedsLongTermCalibration", "Timer");
    }
    else
    {
        disableRTEvent("LE_NeedsLongTermCalibration", "Timer");
    }

if (data.itemAvailable("VA_EnableOnDemandEvent
{
    bool enableOnDemandEventSource = data.VA_EnableOnDemandEventSource.get();
    logTextStream << "\nset onDemand event source enable=" << enableOnDemandEventSource;
    if (enableOnDemandEventSource)
    {
        this->Ex04ServiceLocator_->enableEventSource("ManualCalibrationEventSource");
    }
    else
    {
        this->Ex04ServiceLocator_->disableEventSource(
    }
}
```

name of the logical-event

service-locator of class Ex04

deprecated argument, can be ignored

name of the event-source

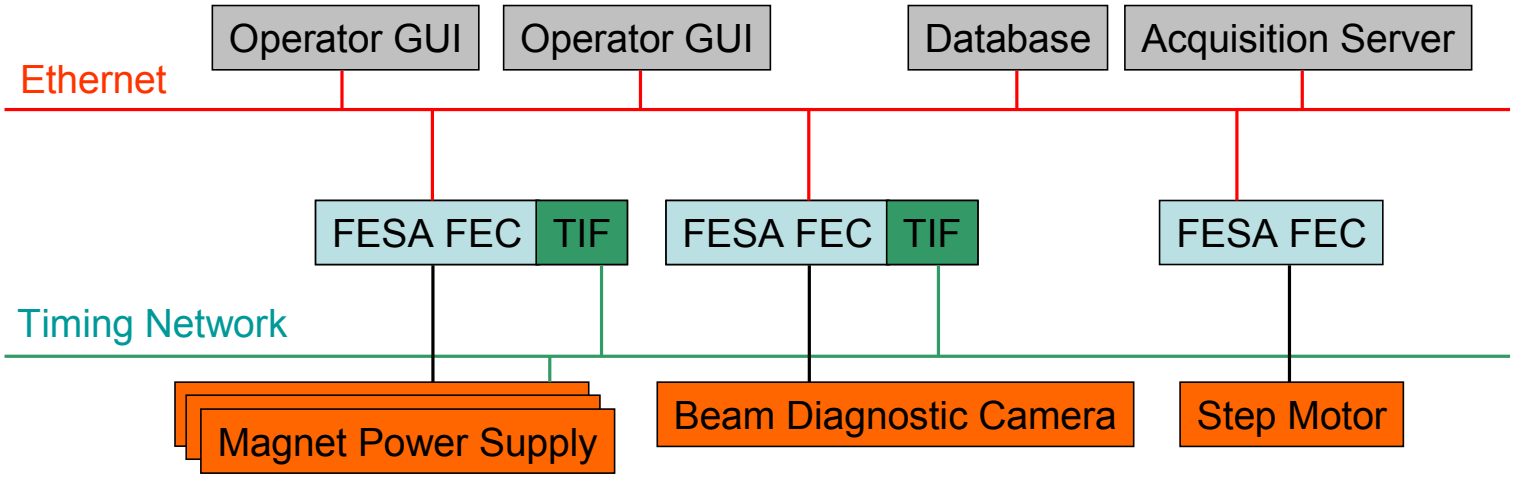
TimerEventSource

- 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]

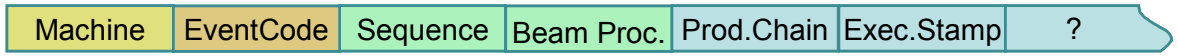
event-configuration	(Timing | Timer | O
name	MyTimerConfig
Timer	(timer-event+)
timer-event	
period	1500
timer-event	
period	3000

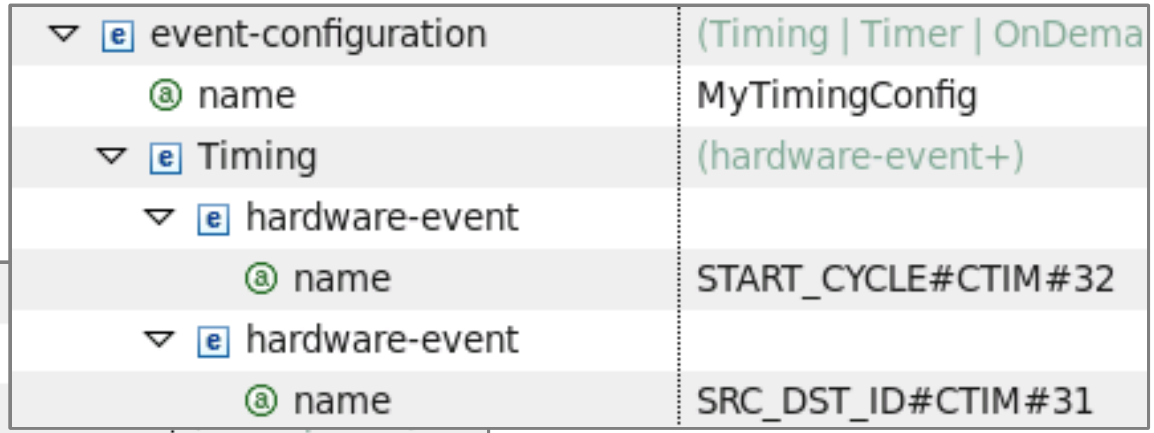
events	
sources	
timer-event-source	
name	Timer

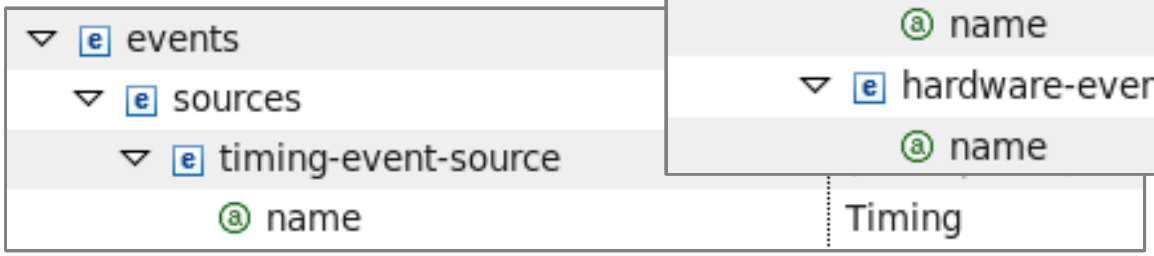
TimingEventSource



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

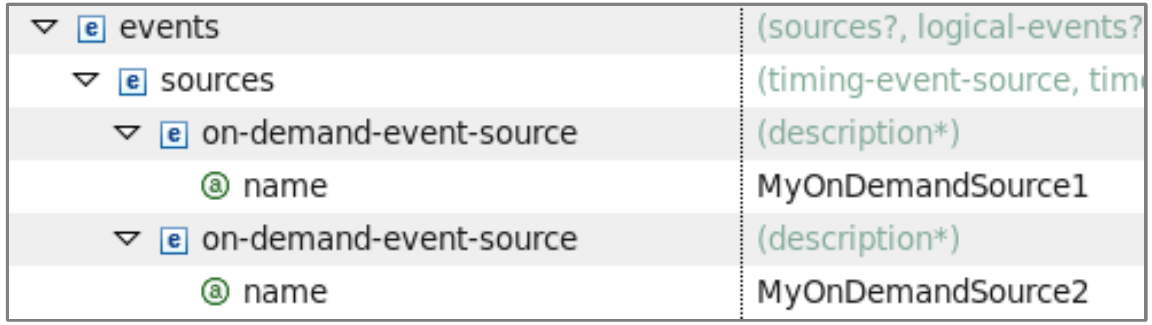






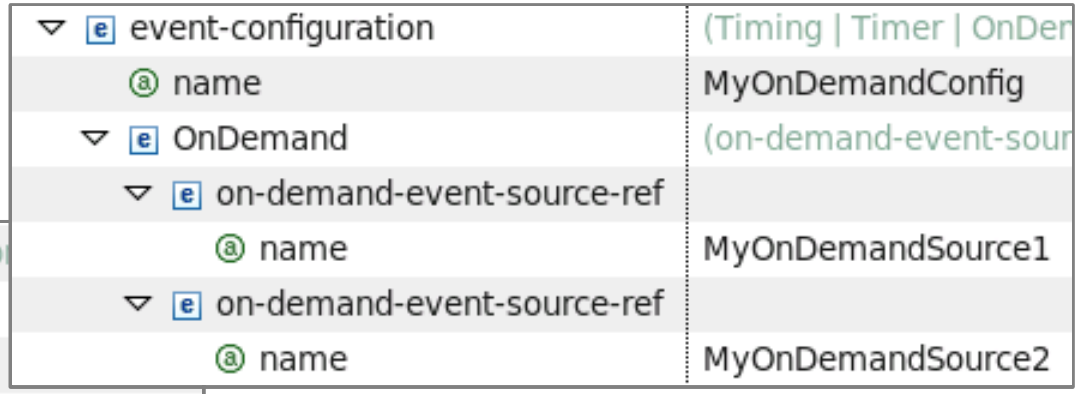
OnDemandEventSource

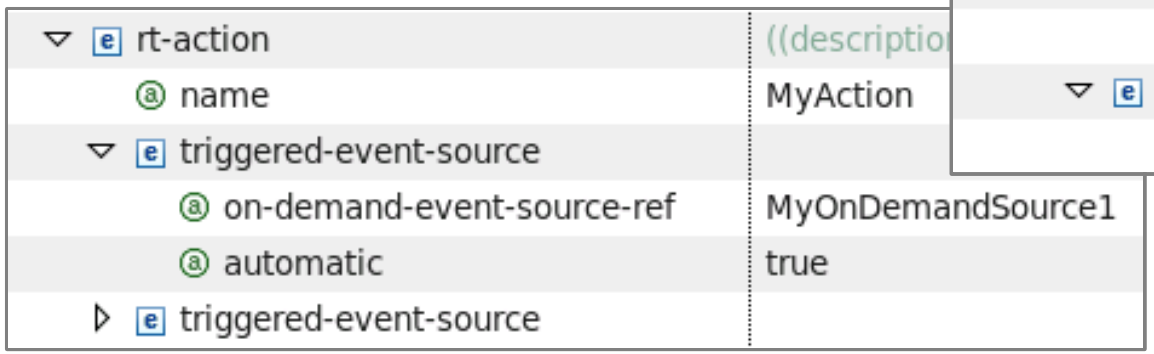
- ServerAction triggers RTAction
- RTAction triggers RTAction
- any number of class-specific instances
- manual or automatic trigger
- payload




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

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





OnSubscriptionEventSource

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

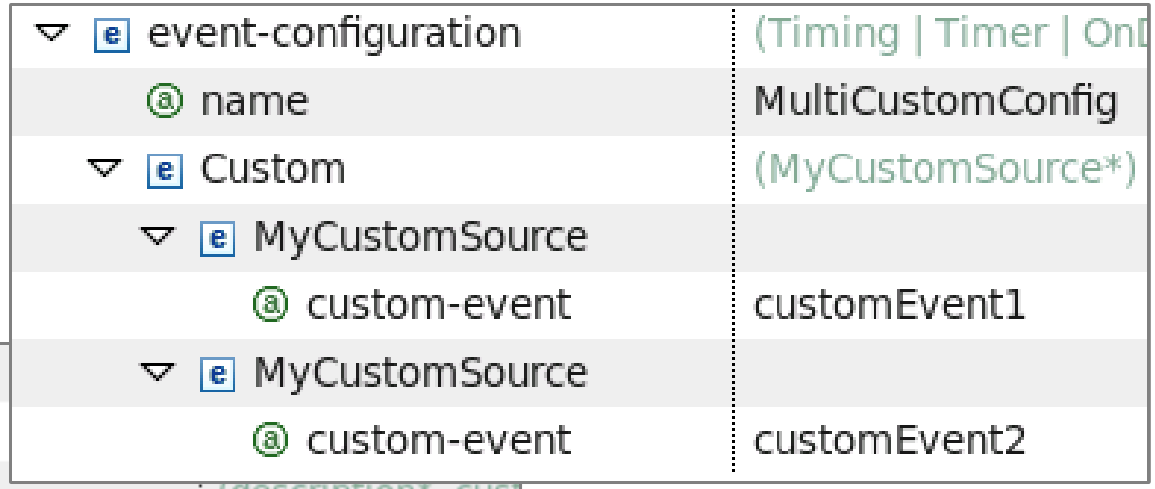
relationship	(association*, composition*,
association	(class-name, class-major-ve
class-name	FFTOnSubscriptionSrc

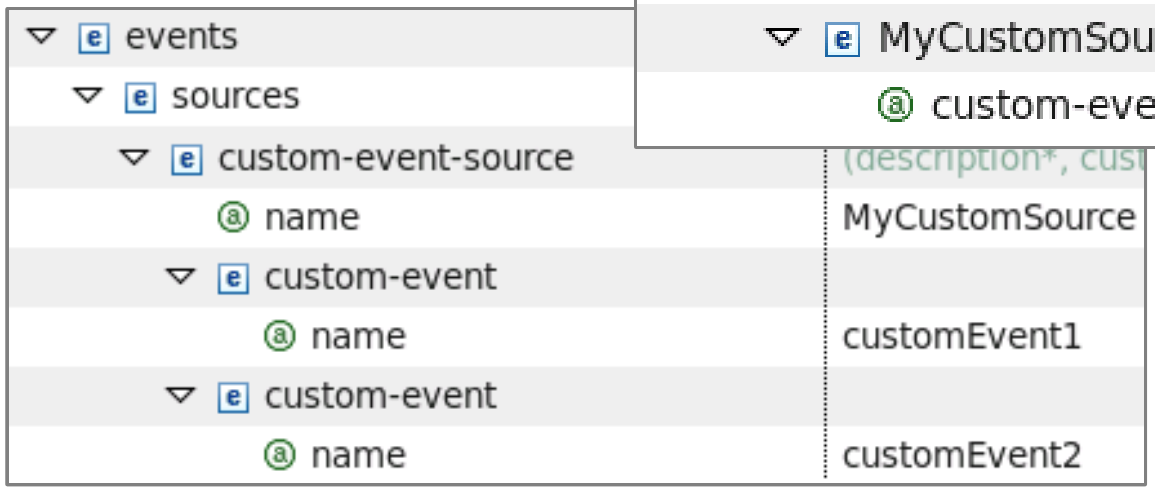
events	
sources	
on-subscription-event-source	(description?)
name	OnSubscription

event-configuration	(Timing | Timer | OnDemand | C
name	MyOnSubscriptionConfig
OnSubscription	(on-subscription-event+)
on-subscription-event	
context	SIS.USER.VACC_12
device	MyDevice1
property	PropertyTriggerOnSubscription
on-subscription-event	
context	NONE
device	MyDevice2
property	PropertyTriggerOnSubscription

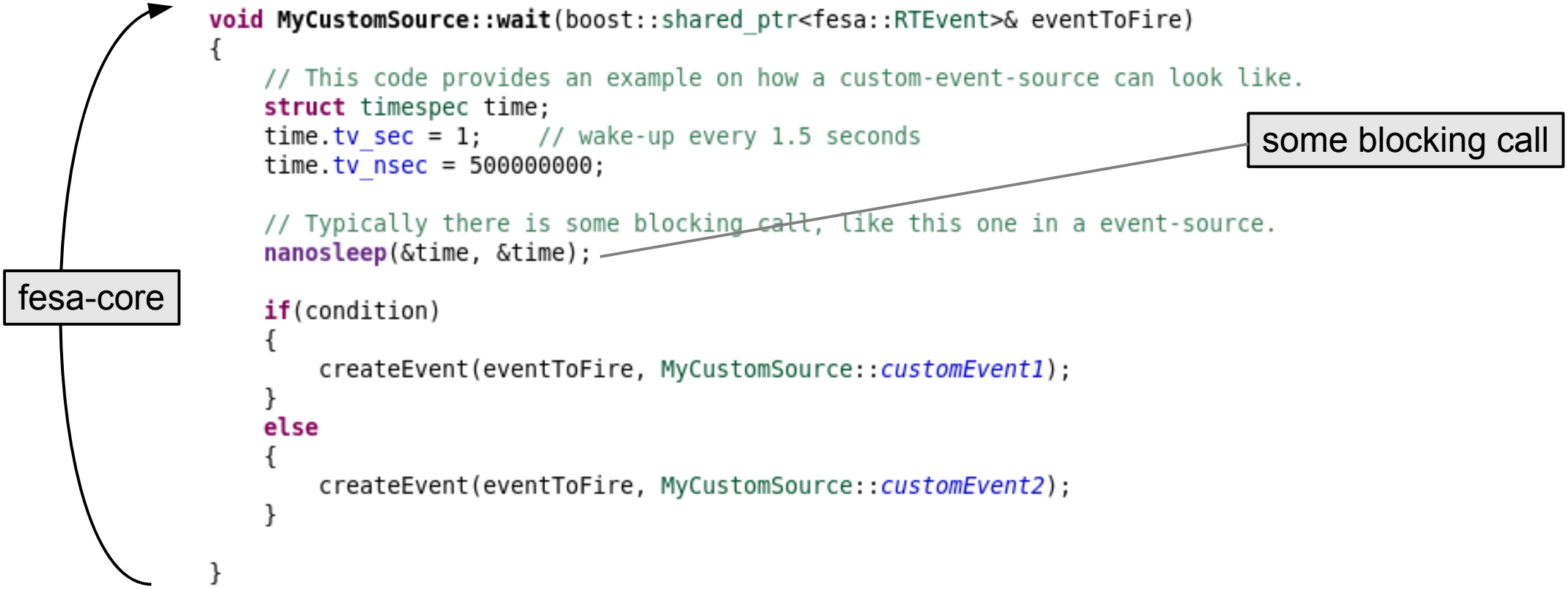
CustomEventSource

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





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);
    }

}
```