Usage of 3rd party code

Class:

Makefile.specific

```
1
2 # Additional compiler flags
3 COMPILER_FLAGS +=
4 LINKER_FLAGS +=
5 # Additional headers (Custom.h Specific.h ...) which need to be installed
6 EXTRA_HEADERS +=
7
8
9
```

Deploy-Unit:

Makefile.specific | Makefile.specific

```
1
2 # Additional libs and flags which are common to the Realtime and Server part
3 COMPILER_FLAGS +=
4 LINKER_FLAGS +=
5
6 # Additional libs and flags which are specific to the Realtime part
7 COMPILER_RT_FLAGS +=
8 LINKER_RT_FLAGS +=
9
10 # Additional libs and flags which are specific to the Server part
11 COMPILER_SERVER_FLAGS +=
12 LINKER_SERVER_FLAGS +=
13
14 # Additional headers (Custom.h Specific.h ...) which need to be released
15 EXTRA_HEADERS +=
16
```

DummyDevice.h

```
1  #ifndef DUMMYDEVICE_H_
2  #define DUMMYDEVICE_H_
3
4  #include <string>
5  #include <stdint.h>
6
7  namespace dummyDevice
8  {
9
10
11 int32_t getDeviceHandle(std::string address);
12
13 void initialize(int32_t deviceHandle);
14
15 double measureVoltage(int32_t deviceHandle);
16
17 void setVoltage(int32_t deviceHandle, double voltage);
18
19 void setMaxVoltage(int32_t deviceHandle, double voltage);
20
21 void setMinVoltage(int32_t deviceHandle, double voltage);
22
23
24 }
25
26
27 #endif /* DUMMYDEVICE_H_ */
28
```

/home/bel/schwinn/lnx/Software/DummyDevice/lib libDummyDevice.a
/include DummyDevice.h
/src

Makefile.specific | Makefile.specific

```
1
2 DUMMY_DEVICE_HOME = /home/bel/schwinn/lnx/Software/DummyDevice
3
4 # Additional compiler flags
5 COMPILER_FLAGS += -I$(DUMMY_DEVICE_HOME)/include
6 LINKER_FLAGS +=
7 # Additional headers (Custom.h Specific.h ...) which need to be installed
8 EXTRA_HEADERS +=
9
10
11
```

Makefile.specific | Makefile.specific

```
1
2 DUMMY_DEVICE_HOME = /home/bel/schwinn/lnx/Software/DummyDevice
3
4 # Additional libs and flags which are common to the Realtime and Server part
5 COMPILER_FLAGS += -I$(DUMMY_DEVICE_HOME)/include
6 LINKER_FLAGS += -L$(DUMMY_DEVICE_HOME)/lib -lDummyDevice
7
8 # Additional libs and flags which are specific to the Realtime part
9 COMPILER_RT_FLAGS +=
10 LINKER_RT_FLAGS +=
11
12 # Additional libs and flags which are specific to the Server part
13 COMPILER_SERVER_FLAGS +=
14 LINKER_SERVER_FLAGS +=
15
16 # Additional headers (Custom.h Specific.h ...) which need to be released
17 EXTRA_HEADERS +=
18
```

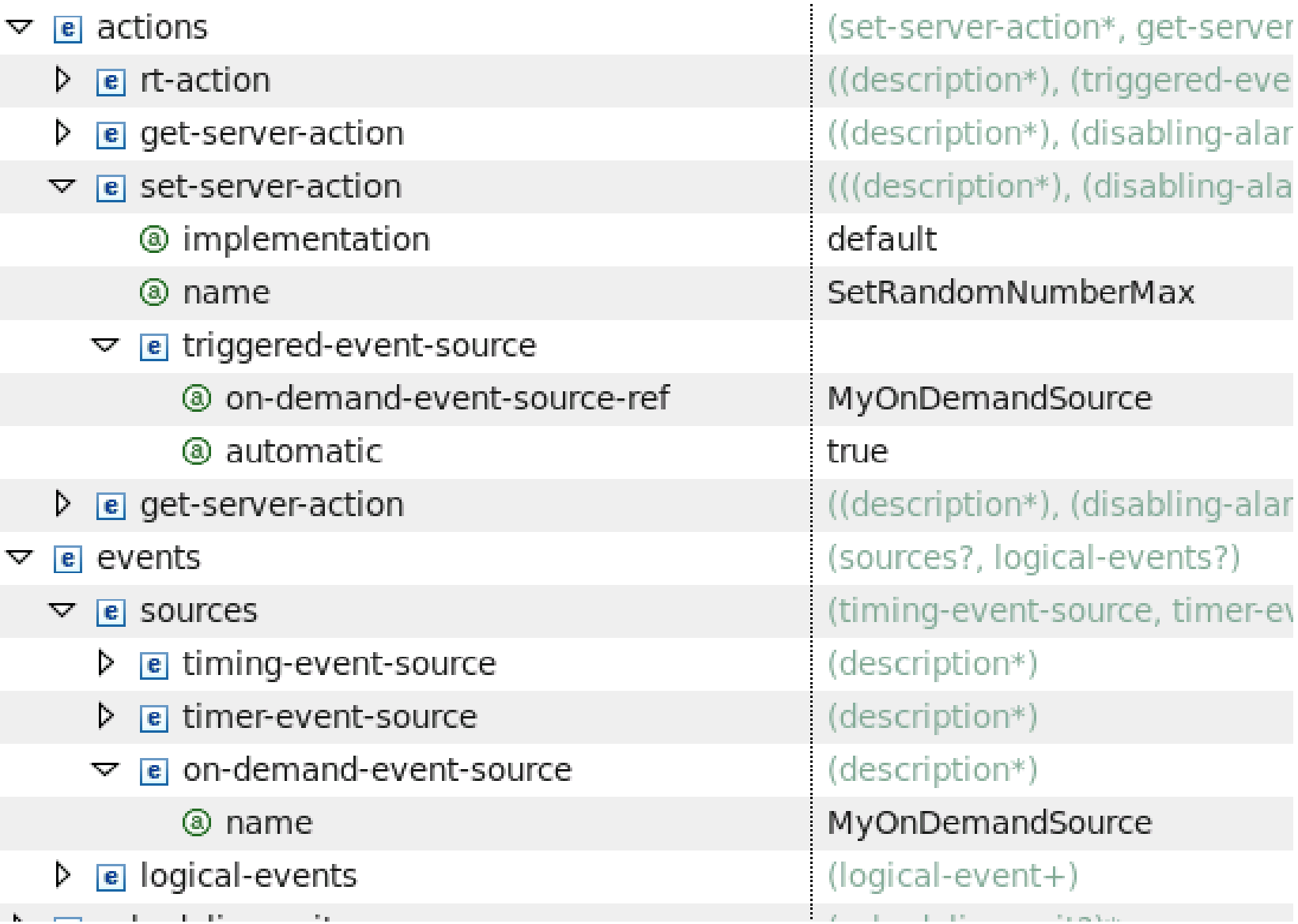
DUMMY_DEVICE_HOME = /common/home/bel/schwinn/lnx/Software/DummyDevice
COMPILER_FLAGS += -I$(DUMMY_DEVICE_HOME)/include
LINKER_FLAGS += -L$(DUMMY_DEVICE_HOME)/lib -lDummyDevice

```
#include <DummyDevice.h>
```

```
void MeasVoltage::execute(fesa::RTEvent* pEvt)
{
    std::vector<Device*>::iterator device;
    for (device = deviceCol_.begin(); device != deviceCol_.end(); device++ )
    {
        double measVoltage;
        try
        {
            //int32_t hwDeviceHandle = dummyDevice::getDeviceHandle((*device)->HardwareDeviceName.get()->hostName);
            int32_t hwDeviceHandle = dummyDevice::getDeviceHandle("myDeviceAdress");
            measVoltage = dummyDevice::measureVoltage(hwDeviceHandle);
            (*device)->measVoltage.set(measVoltage,pEvt->getMultiplexingContext());
        }
        catch(std::string ex)
        {
            std::cout << "An error happened in the device: " <<  ex << std::endl;
            return;
        }

        std::cout << "Voltage measured successfully! Voltage: " <<  measVoltage << std::endl;
    }
}
```

Quick crash-course: The OnDemandEventSource

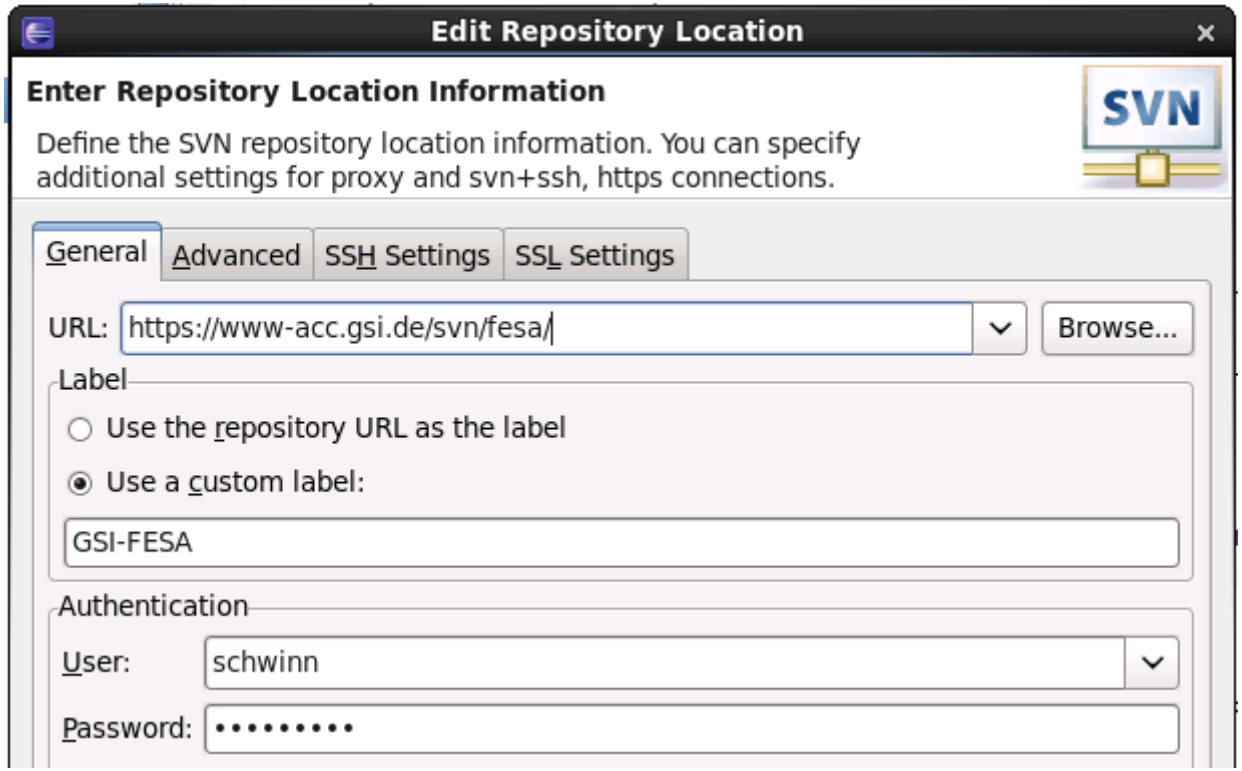


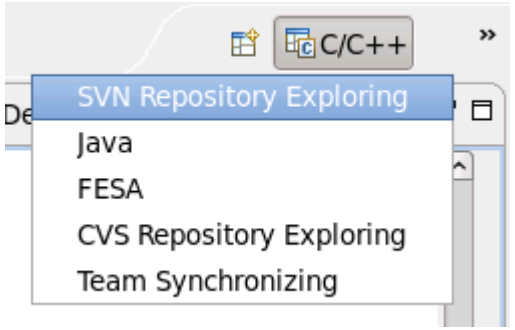
Mission

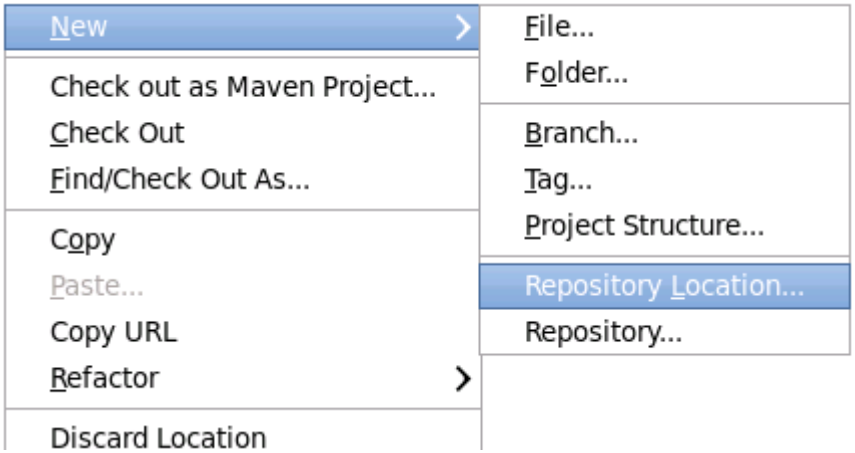
- Build a FESA-class in order to control an instance of a “DummyDevice”
- Keep all device-interaction on the RT-side
- Measure the Voltage of “DummyDevice” once a second.
- Allow clients to display the measured data, to set a voltage and to set min/max limits
- Use an OnDemandEventSource in order to trigger RT-Actions by Set-Server-Actions
- Use a custom-server-action to throw an exception if the set-value is out of range
- Hint: You have to initialize the hw-device before usage! Best place for that is the RTDeviceClass

On any problems: fesa-support@gsi.de

SVN - Usage

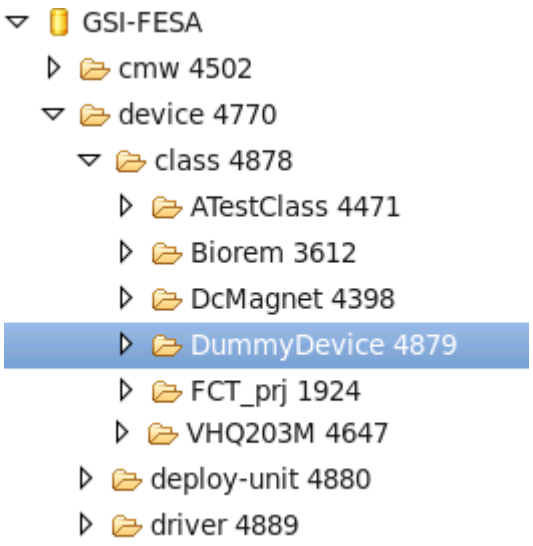

Edit Repository Location
Enter Repository Location Information
Define the SVN repository location information. You can specify additional settings for proxy and svn+ssh, https connections.
SVN
General
Advanced
SSH Settings
SSL Settings
URL: https://www-acc.gsi.de/svn/fesa/
Browse...
Label
Use the repository URL as the label
Use a custom label:
GSI-FESA
Authentication
User: schwinn
Password: ••••••••••

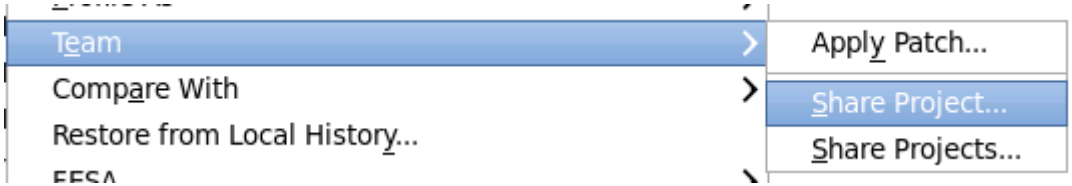

C/C++
SVN Repository Exploring
Java
FESA
CVS Repository Exploring
Team Synchronizing

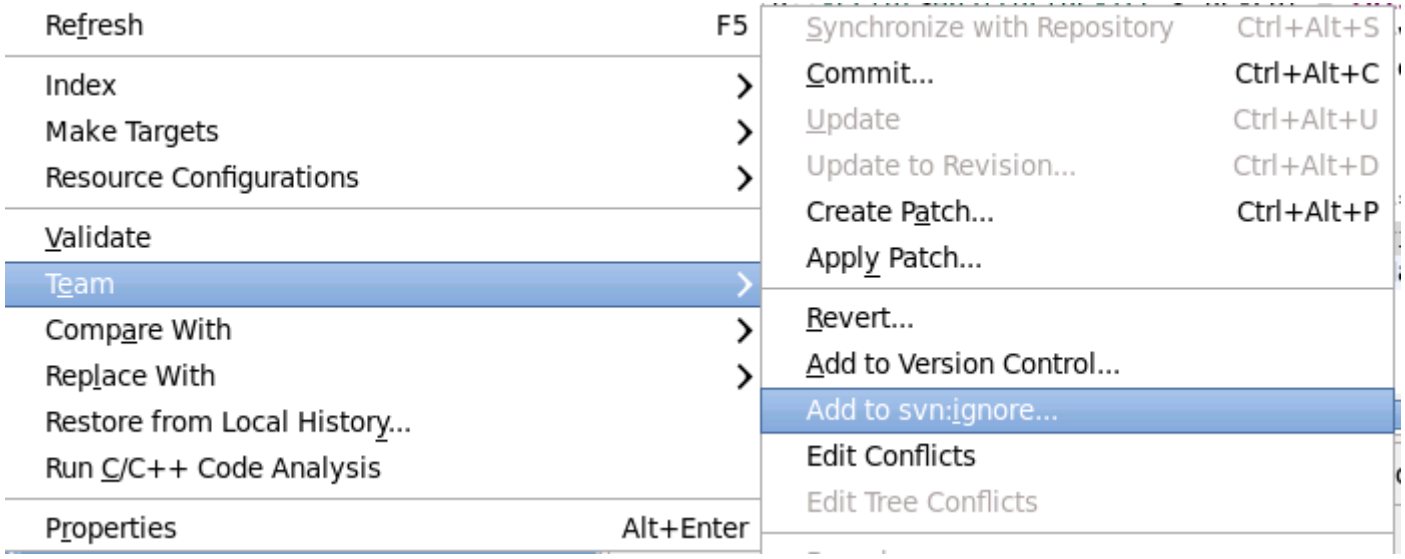

New
Check out as Maven Project...
Check Out
Find/Check Out As...
Copy
Paste...
Copy URL
Refactor
Discard Location
File...
Folder...
Branch...
Tag...
Project Structure...
Repository Location...
Repository...

SVN - Usage

https://www-acc.gsi.de/viewvc/view/fesa/device/







Thank you for your listening !!