# JavaIOC: Support

Marty Kraimer
EPICS Collaboration Meeting
DESY April 23-27 2007

# Overview

- Two flavours of support
  - Support – record support is just support
  - linkSupport – Can have configStructure
- Every field can optionally have support
- Support found via Java Reflection
- Support does some combination of
  - Implement some function
  - Call other support

# Example Support

- doubleRecord is the support for:
  - record types
    - double
    - aiDouble
    - aoDouble
  - structures like the above record types
    - embeded in powerSupply
  - Can also be used by other records/structures

# doubleRecord support

- Constructor is:
  - DoubleImpl(DBStructure dbStructure)
- Requires a value field
  - Field named “value” in structure
  - Passed to method setField(DBField value)
- Calls support for fields named
  - input, doubleAlarm, output, linkArray
  - If field or support not found just ignore.

# doubleCommon.xml

```
<?xml version="1.0" ?>
<DBDefinition>
    <field name = "value" type = "double" >
        <property name = "alarm" associatedField = "alarm" />
        <property name = "units" associatedField = "units" />
        <property name = "displayLimit" associatedField = "displayLimit" />
        <property name = "timeStamp" associatedField = "/timeStamp" />
    </field>
    <field name = "input" type = "link" />
    <field name = "doubleAlarm" type = "link" supportName = "doubleAlarm" />
    <field name = "output" type = "link" />
    <field name = "linkArray" type = "array"
         elementType = "structure" supportName = "linkArray" />
    <field name = "units" type = "string" />
    <field name = "displayLimit"
         type = "structure" structureName = "doubleLimit" />
</DBDefinition>
```

# doubleRecord.xml

```
<?xml version="1.0" ?>
<DBDefinition>

<structure name = "double" supportName = "doubleRecord" >
    <include href = "doubleCommon.xml" />
</structure>

<recordType name = "double" supportName = "doubleRecord" >
    <include href = "common.xml" />
    <include href = "doubleCommon.xml" />
</recordType>

<support name = "doubleRecord"
   factoryName = "org.epics.ioc.recordSupport.DoubleFactory" />


</DBDefinition>
```

# aiDoubleCommon.xml

```
<?xml version="1.0" ?>
<DBDefinition>
    <field name = "value" type = "double" >
        <property name = "alarm" associatedField = "alarm" />
        <property name = "units" associatedField = "units" />
        <property name = "displayLimit"
             associatedField = "displayLimit" />
        <property name = "timeStamp"
             associatedField = "/timeStamp" />
    </field>
    <field name = "input" type = "structure"
         structureName = "linearConvertInput"/>
    <field name = "units" type = "string" />
    <field name = "doubleAlarm" type = "link"
         supportName = "doubleAlarm" />
    <field name = "displayLimit"
         type = "structure" structureName = "doubleLimit" />
    <field name = "linkArray" type = "array"
         elementType = "structure" supportName = "linkArray" />
</DBDefinition>
```

# aiDoubleRecord.xml

```
<?xml version="1.0" ?>
<DBDefinition>

<structure name = "aiDouble" supportName = "doubleRecord" >
    <include href = "aiDoubleCommon.xml" />
</structure>


<recordType name = "aiDouble" supportName = "doubleRecord" >
    <include href = "common.xml" />
    <include href = "aiDoubleCommon.xml" />
</recordType>

</DBDefinition>
```

# aoDoubleCommon.xml

```
<?xml version="1.0" ?>
<DBDefinition>
    <field name = "value" type = "double" >
        <property name = "alarm" associatedField = "alarm" />
        <property name = "units" associatedField = "units" />
        <property name = "displayLimit" associatedField = "displayLimit" />
        <property name = "controlLimit" associatedField = "controlLimit" />
        <property name = "timeStamp" associatedField = "/timeStamp" />
    </field>
    <field name = "input" type = "link" />
    <field name = "output" type = "structure" structureName = "linearConvertOutput"/>
    <field name = "units" type = "string" />
    <field name = "doubleAlarm" type = "link" supportName = "doubleAlarm" />
    <field name = "displayLimit"
         type = "structure" structureName = "doubleLimit" />
    <field name = "linkArray" type = "array"
         elementType = "structure" supportName = "linkArray" />
</DBDefinition>
```

# aoDoubleRecord.xml

```
<?xml version="1.0" ?>
<DBDefinition>

<structure name = "aoDouble" supportName = "doubleRecord" >
    <include href = "aoDoubleCommon.xml" />
</structure>

<recordType name = "aoDouble" supportName = "doubleRecord" >
    <include href = "common.xml" />
    <include href = "aoDoubleCommon.xml" />
</recordType>

</DBDefinition>
```

# DoubleFactory.java

```
public class DoubleFactory {
    public static Support create(DBStructure dbStructure) {
        return new DoubleImpl(dbStructure);
    }

    static private class DoubleImpl extends AbstractSupport
    implements SupportProcessRequestor
    { ... }
}
```

# DoubleImpl

- Extends AbstractSupport: overrides
  - initialize,start,stop,uninitialize
  - process
  - setField
- Implements SupportProcessRequestor
  - SupportProcessDone
- Source is 253 lines including javaDoc
  - aiRecord.c is 445 lines no doc.

# DoubleImpl Cont.

```
private static String supportName = "doubleRecord";
private DBStructure dbStructure;
private PVStructure pvStructure;
private DBField valueDBField = null;
private Support inputSupport = null;
private Support doubleAlarmSupport = null;
private Support outputSupport = null;
private Support linkArraySupport = null;
private SupportProcessRequestor supportProcessRequestor = null;
private ProcessState processState = ProcessState.inputSupport;
private RequestResult finalResult = RequestResult.success;

private DoubleImpl(DBStructure dbStructure) {
    super(supportName,dbStructure);
    this.dbStructure = dbStructure;
    pvStructure = dbStructure.getPVStructure();
}

private enum ProcessState {
    inputSupport,
    doubleAlarmSupport,
    outputSupport,
    linkArraySupport
}
```

# DoubleImpl.initialize

```
public void initialize() {
    if(!super.checkSupportState(
        SupportState.readyForInitialize,supportName)) return;
    SupportState supportState = SupportState.readyForStart;
    Structure structure = (Structure)pvStructure.getField();
    DBField[] dbFields = dbStructure.getFieldDBFields();
    int index;
    if(valueDBField==null) {
        index = structure.getFieldIndex("value");
        if(index<0) {
            super.message("no value field", MessageType.error);
            return;
        }
        valueDBField = dbFields[index];
    }
    index = structure.getFieldIndex("input");
    if(index>=0) {
        inputSupport = dbFields[index].getSupport();
    }
    ... for other supported fields
    if(inputSupport!=null) {
        inputSupport.setField(valueDBField);
        inputSupport.initialize();
        supportState = inputSupport.getSupportState();
        if(supportState!=SupportState.readyForStart) return;
    }
    ... for other supported fields
}
```

# DoubleImpl Cont.

- initialize,start,stop,uninitialize
  - call lower level support
- initialize – initialization within record.
- start – connect to hardware or other records
- stop - disconnects from hardware or other records
- uninitialize – get ready to initialize

# DoubleImpl.process

```
public void process(SupportProcessRequestor supportProcessRequestor) {
    if(!super.checkSupportState(SupportState.ready,"process")) {
        supportProcessRequestor.supportProcessDone(RequestResult.failure);
        return;
    }
    if(supportProcessRequestor==null) {
        throw new IllegalStateException("supportProcessRequestor is null");
    }
    this.supportProcessRequestor = supportProcessRequestor;
    finalResult = RequestResult.success;
    if(inputSupport!=null) {
        processState = ProcessState.inputSupport;
        inputSupport.process(this);
    } else if(doubleAlarmSupport!=null) {
        processState = ProcessState.doubleAlarmSupport;
        doubleAlarmSupport.process(this);
    } ... for other supported fields
    } else {
        supportProcessRequestor.supportProcessDone(RequestResult.success);
    }
}
```

# DoubleImpl.setField

```
public void setField(DBField dbField) {
    valueDBField = dbField;
}
```

# DoubleImpl.supportProcessDone

```
public void supportProcessDone(RequestResult requestResult) {
    if(requestResult.compareTo(finalResult)>0) {
        finalResult = requestResult;
    }
    switch(processState) {
    case inputSupport:
        if(doubleAlarmSupport!=null) {
            processState = ProcessState.doubleAlarmSupport;
            doubleAlarmSupport.process(this);
            return;
        }
    case doubleAlarmSupport:
        if(outputSupport!=null) {
            processState = ProcessState.outputSupport;
            outputSupport.process(this);
            return;
        }
    case outputSupport:
        if(linkArraySupport!=null) {
            processState = ProcessState.linkArraySupport;
            linkArraySupport.process(this);
            return;
        }
    case linkArraySupport:
        supportProcessRequester.supportProcessDone(finalResult);
        return;
    }
}
```