JavaIOC

Marty Kraimer
ICALEPCS 2007

JavaIOC – What Is It?

- IOC - Input/Output Controller
- Has a “smart” real time database
  - Record instances can be processed.
  - Records can be accessed via Channel Access.
  - Records can link to hardware, other records, etc.
- Has functionality like an EPICS IOC but
  - Structured data
  - Better Array data
  - Generic Support
  - Written in Java

Features

- PV (Process Variable) Database
- DBD - Database Definition Database
- DB – Runtime Database of record instances
- XML Parsers for DBD and DB
- Record Processing
- Record Scanning: Periodic and Event
- Record Monitoring
- Channel Access
- Generic structure/recordTypes and support

PV Database

- Field types
  - Primitive: all Java primitives except char
    - boolean, byte, short, int, long, float, double
  - string: implemented as a Java String
  - structure: has fields. Each field can be any type
  - array: has elementType which can be any type
- Complex Structures fully supported
- Introspection and Data interfaces: Provide access to any field.

PV Introspection

- Introspection Interfaces, i.e. no data
  - Field: Methods: getType, getFieldName, ...
  - Array: extends Field: Method: getElementType
  - Structure: extends Field
    - Methods: getFields, getStructureName
- FieldFactory
  - Implements introspection interfaces
  - Can be extended but probably not necessary

PV Data Interfaces

- PVField: Base for data: Methods: getField, ...
  - PVBoolean,...,PVString : Methods: get,put
  - PVArray: Base for array data interfaces
    - PVBooleanArray,...,PVArrayArray : Methods get,put
  - PVStructure provides access to a structure.
  - PVRecord provides access to a record instance.
- PVDataFactory
  - Default implementation.
  - Any PVField can be replaced. Often useful
- ConvertFactory: Convert between data types

org.epics.ioc.pv

- Self Contained Java package for PV Data
- Can be used for other than JavaIOC
  - Example: Channel Access
    Implements CD Data by using PV Data
- Designed for use by other Java Facilities
- Via structures could support
  - Vector/Matrix Data
  - Image Data
  - etc.

XML Parsers

- DBD - Database Definitions
  - structure
  - recordType - A top level structure
  - create – defines factory that replaces default data implementation.
  - support – defines a factory that implements support for a field.
- DB – Database of record Instances
- Macro Substitution and Include

PV Naming

- EPICS pvname is <record>.field
- JavaIOC is <record>.name.name. ...
  - name can be field name or a property
- Some examples
  - <record>.value
<record>.value.display.limit.low
<powerSupply>.power.value
<psArray>.supply[0].power.value

Record Processing

- A record instance is basic process unit
  - It is locked during any processing or I/O
  - RecordProcess is master. It calls support for record instance.
  - Support modules implement semantics
  - ANY field can optionally have support
  - Record instance must have support

Record Scanning

- Scan
  - Types: passive, periodic, event
  - Priority Uses Java priorities
  - Threads created as required
- Periodic
  - Arbitrary rate: minPeriod,deltaPeriod
- Event
  - Based on eventName
  - Replaces EPICS event and I/O Inter

Support

- Implements Record Processing Semantics
- Each record instance has support
- Each field can optionally have support
- Support can call other support
- Example: generic – support for a structure
  - Calls support for each field that has support
  - Each support must finish before next is called
  - Supports a recordType or a structure

JavaIOC Data Model

- Simple: All related data appears together in a structure
- Intended for Client tools and for Support
- Field name can be a property name
  - A null structure(no fields and no support) is not a property.
  - If a structure has a field named “value” than every other field is a property unless it is a null structure.

Finding Properties

- PVField provides method findProperty
  - PVField findProperty(String fieldName);
- Two examples for fieldName are:
  - “value”
  - “supply[0].power.value”
- If pvField refers to the value field than
  - “display”
  - “display.limit.low”

structure/recordType double

- Next slide shows structure
- Follows JavaIOC Data Model
- structure double can:
  - Just holds data
  - Be an input or output or both
  - Can be embeded in other recordTypes
  - Is a building block for “device” records
- recordType double adds scan field

double.xml

```
<structure name = “double” supportName = “generic” >
  <field name = "value" type = "double" />
  <field name = "alarm" type = "structure"/>
  <field name = "timeStamp" type = "structure" />
  <field name = "input" type = "structure" />
  <field name = "valueAlarm" type = "structure" />
  <field name = "output" type = "structure" />
  <field name = "display" type = "structure" />
  <field name = "control" type = "structure" />
  <field name = "history" type = "structure" />
</structure>
```

Fields in double

- value has type double
- All other fields are a null structure
  - A record instance can override
  - If not overridden than NOT a property
- Following two slides show example
  - The definition of display
  - An example record instance

display.xml

```
<structure name = "display">
    <field name = "description" type = "string" />
    <field name = "format" type = "string" />
    <field name = "units" type = "string" />
    <field name = "resolution" type = "int" />
    <field name = "limit" type = "structure"
        structureName = "doubleLimit" />
</structure>
```

record instance

example.value will have property display.

```
<record name = "example" type = “double” >
    <display structureName = “display”>
        <units>volts</units>
        <limit>
            <low>0.0</low>
            <high>10.0</high>
        </limit>
    </display>
</structure>
```

Support Overview

- Implements record processing semantics
- Each record instance has support
- Each field of a record instance can optionally have support
- Support is as generic as possible
  - During initialization look for required fields
  - If required fields not found don’t start
- generic is often the support

Analog Input Example

- The recordType is double
- input has structureName=linearConvert
  - This has fields value, input, and linearConvert
  - The support is generic which calls
    - pdrvInt32Input - accesses hardware
    - linearConvertInput

Analog Input Instance

```
ai              recordType=double support=generic
  value         A double that is set by linearConvertInput
  input         structureName=linearConvertInput
                support=generic
     value           An int that is set by pdrvInt32Input
     input           supportName=pdrvInt32Input
     linearConvert supportName = linearConvertInput
```

When ai is processed

```
support for ai calls support for ai.input
  support for ai.input (generic)
     calls support for ai.input.input (pdrvInt32Input)
        gets an int value and puts result in ai.input.value
     calls support for ai.input.linearConvert
        gets ai.input.value, converts it, and puts ai.value
```

powerSupply Example

- This is an example of a “device” record
- Only new support is powerSupplyCurrent
  - It gets power.value and voltage.value
  - From these it computes the current
  - Puts the current into current.value

PowerSupply DBD

```
<structure name = “powerSupply”
    supportName = “generic”>
  <field name = "power" type = "structure"
      structureName = "double" />
  <field name = "voltage" type = "structure"
      structureName = "double" />
  <field name = "current" type = "structure"
      structureName = "double" />
</structure>
```

powerSupplyArray

The following defines a recordType that can hold an array of powerSupply

```
<recordType name = "powerSupplyArray"
     supportName = "generic" >
  <include href = "common.xml" />
  <!-- each element must be a powerSupply -->
  <field name = "supply" type = "array”
      elementType = “structure”
      supportName = “generic” />
  <field name = "alarm" type = "structure"/>
  <field name = "timeStamp" type = "structure" />
</recordType>
```

Finding Fields

- A client could ask for
  - ai.value
  - ai.input.value
  - ps.power.value
  - ps.current.value
  - ps.voltage.value
  - psArray.supply[0].power.value
  - psArray.supply[1].current.value