EPICS Database Principles

Andrew Johnson
APS Engineering Support Division

October 17th, 2006
SNS EPICS Training
Outline

- Records
- Fields and field types
- Record Scanning
- Input and Output record types
- Links, link address types
- Connecting records together
- Protection mechanisms
- Alarms, deadbands, simulation and security
A control system using EPICS will contain one or more IOCs
Each IOC loads one or more Databases telling it what to do
A Database is a collection of Records of various types
A Record is an object with:
  - A unique name
  - A behavior defined by its record type (class)
  - Controllable properties (fields)
  - Optional associated hardware I/O (device support)
  - Links to other records
Record Activity

- Records are active — they can do things:
  - Get data from other records or from hardware
  - Perform calculations
  - Check values are in range & raise alarms
  - Put data to other records or to hardware
  - Activate or disable other records
  - Wait for hardware signals (interrupts)
- What a record does depends upon its record type and the settings of its fields
- No action occurs unless a record is processed
How is a Record implemented?

- A ‘C’ structure with a data member for each record field
  - All records start with a standard set of fields (dbCommon) that the system needs, including pointers to record type information
- A record definition within a database provides
  - Record name
  - The record’s type
  - Values for each design field
- A record type provides
  - Definitions of all the fields
  - Code which implements the record behaviour
- New record types can be added to an application as needed
One view of a Record

<table>
<thead>
<tr>
<th>DESC</th>
<th>STRING</th>
<th>Descriptor</th>
<th>Temperature Demand</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGG</td>
<td>STRING</td>
<td>Access Security Group</td>
<td></td>
</tr>
<tr>
<td>SCAN</td>
<td>MENU</td>
<td>Scan Mechanism</td>
<td>1 second</td>
</tr>
<tr>
<td>PINI</td>
<td>MENU</td>
<td>Process at t0cinit</td>
<td>NO</td>
</tr>
<tr>
<td>PHAS</td>
<td>INTEGER</td>
<td>Scan Phase</td>
<td>30</td>
</tr>
<tr>
<td>EVNT</td>
<td>INTEGER</td>
<td>Event Number</td>
<td>30</td>
</tr>
<tr>
<td>TSEL</td>
<td>INTEGER</td>
<td>Time Stamp Event</td>
<td>30</td>
</tr>
<tr>
<td>TSEL</td>
<td>INLINK</td>
<td>Time Stamp Link</td>
<td></td>
</tr>
<tr>
<td>DTPY</td>
<td>DEVICE</td>
<td>Device Type</td>
<td>Soft Channel</td>
</tr>
<tr>
<td>OUT</td>
<td>OUTLINK</td>
<td>Output Specification</td>
<td>Form</td>
</tr>
<tr>
<td>DISV</td>
<td>INTEGER</td>
<td>Disable Value</td>
<td>M</td>
</tr>
<tr>
<td>SDIS</td>
<td>INLINK</td>
<td>Scanning Disable</td>
<td></td>
</tr>
<tr>
<td>ACKT</td>
<td>MENU</td>
<td>Alarm Ack Transient</td>
<td>YES</td>
</tr>
<tr>
<td>DISS</td>
<td>MENU</td>
<td>Disable Alarm Severity</td>
<td>NO_ALARM</td>
</tr>
<tr>
<td>PRIO</td>
<td>MENU</td>
<td>Scheduling Priority</td>
<td>LOW</td>
</tr>
<tr>
<td>UDF</td>
<td>INTEGER</td>
<td>Undefined</td>
<td>M</td>
</tr>
<tr>
<td>FUNK</td>
<td>FWLINK</td>
<td>Forward Process Link</td>
<td>Form</td>
</tr>
<tr>
<td>VAL</td>
<td>REAL</td>
<td>Desired Output</td>
<td></td>
</tr>
<tr>
<td>ORIC</td>
<td>REAL</td>
<td>Output Rate of Change</td>
<td>30</td>
</tr>
<tr>
<td>DOL</td>
<td>INLINK</td>
<td>Desired Output Loc</td>
<td>UserDemand NPP NMS</td>
</tr>
<tr>
<td>OMSL</td>
<td>MENU</td>
<td>Output Mode Select</td>
<td>supervisory</td>
</tr>
<tr>
<td>OIF</td>
<td>MENU</td>
<td>Out Full/Incremental</td>
<td>FULL</td>
</tr>
</tbody>
</table>
A graphical view of a Record
Another graphical view of a Record

The small CapFast symbol for an Analogue Output record
The IOC's view

The full .db file entry for an Analogue Output Record

```
record(ao,"DemandTemp") {
  field(DESC,"Temperature")
  field(ASG,""")
  field(SCAN,"Passive")
  field(PINI,"NO")
  field(PHAS,"0")
  field(EVNT,"0")
  field(DTYP,"VMIC 4100")
  field(DISV,"1")
  field(SDIS,""")
  field(DISS,"NO_ALARM")
  field(PRIO,"LOW")
  field(FLNK,""")
  field(OUT,"#C0 S0")
  field(OROC,"0.0e+00")
  field(DOL,""")
  field(OMSL,"supervisory")
  field(OIF,"Full")
  field(PREC,"1")
  field(LINR,"NO CONVERSION")
  field(EGUF,"100")
  field(EGUL,"0")
  field(EGU,"Celcius")
  field(DRVH,"100")
  field(DRVL,"0")
  field(HOPR,"80")
  field(LOPR,"10")
  field(HIHI,"0.0e+00")
  field(LOLO,"0.0e+00")
  field(HIGH,"0.0e+00")
  field(LOW,"0.0e+00")
  field(HHSV,"NO_ALARM")
  field(LLSV,"NO_ALARM")
  field(HSV,"NO_ALARM")
  field(LSV,"NO_ALARM")
  field(HYST,"0.0e+00")
  field(ADEL,"0.0e+00")
  field(MDEL,"0.0e+00")
  field(SIOL,""")
  field(SIML,""")
  field(SIMS,"NO_ALARM")
  field(IVOA,"Continue normally")
  field(IVOV,"0.0e+00")
}
```

This shows only the design fields; there are other fields which are used only at run-time
Fields are for...

- Defining
  - What causes a record to process
  - Where to get/put data from/to
  - How to turn raw I/O data into a numeric engineering value
  - Limits indicating when to report an alarm
  - When to notify value changes to a client monitoring the record
  - A Processing algorithm
  - Anything else which needs to be set for each record of a given type

- Holding run-time data
  - Input or output values
  - Alarm status, severity and acknowledgments
  - Processing timestamp
  - Other data for internal use
Field types — fields can contain:

- Integers
  - char, short or long
  - signed or unsigned
- Floating-point numbers
  - float or double
- Fixed length strings
  - maximum useful length is 40 characters
- Enumerated/menu choices
  - select one of up to 16 strings
  - stored as a short integer
- Arrays of any of the above types
- Links
  - to other records in this or other IOCs
  - to hardware signals (device support)
  - provide a means of getting or putting a value
- Other private data
  - not accessible remotely
All Records have these design fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>60 Character unique name (using more than 28 can cause problems)</td>
</tr>
<tr>
<td>DESC</td>
<td>28 Character description</td>
</tr>
<tr>
<td>ASG</td>
<td>Access security group</td>
</tr>
<tr>
<td>SCAN</td>
<td>Scan mechanism</td>
</tr>
<tr>
<td>PHAS</td>
<td>Scan order (phase)</td>
</tr>
<tr>
<td>PINI</td>
<td>Process at IOC initialization?</td>
</tr>
<tr>
<td>PRIO</td>
<td>Scheduling priority</td>
</tr>
<tr>
<td>SDIS</td>
<td>Scan disable input link</td>
</tr>
<tr>
<td>DISV</td>
<td>Scan disable value</td>
</tr>
<tr>
<td>DISS</td>
<td>Disabled severity</td>
</tr>
<tr>
<td>FLNK</td>
<td>Forward link</td>
</tr>
</tbody>
</table>
All Records have these Run-time fields

\begin{itemize}
\item \textbf{PROC} \hspace{1em} \textit{Force processing}
\item \textbf{PACT} \hspace{1em} \textit{Process active}
\item \textbf{STAT} \hspace{1em} \textit{Alarm status}
\item \textbf{SEVR} \hspace{1em} \textit{Alarm severity}
\item \textbf{TPRO} \hspace{1em} \textit{Trace processing}
\item \textbf{UDF} \hspace{1em} \textit{Non-zero if record value undefined}
\item \textbf{TIME} \hspace{1em} \textit{Time when record was last processed}
\end{itemize}
Record Scanning

- **SCAN** field is a menu choice from
  - Periodic — 0.1 seconds .. 10 seconds
  - I/O Interrupt (if device supports this)
  - Soft event — **EVNT** field
  - Passive (default)
- The number in the **PHAS** field allows processing order to be set within a scan
  - Records with **PHAS**=0 are processed first
  - Then those with **PHAS**=1, **PHAS**=2 etc.
- Records with **PINI**=YES are processed once at startup
- **PRIO** field selects Low/Medium/High priority for Soft event and I/O Interrupts
- A record is also processed whenever any value is written to its **PROC** field
**Input records often have these fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INP</td>
<td>Input link</td>
</tr>
<tr>
<td>DTYP</td>
<td>Device type</td>
</tr>
<tr>
<td>RVAL</td>
<td>Raw data value</td>
</tr>
<tr>
<td>VAL</td>
<td>Engineering value</td>
</tr>
<tr>
<td>LOPR</td>
<td>Low operator range</td>
</tr>
<tr>
<td>HOPR</td>
<td>High operator range</td>
</tr>
</tbody>
</table>
### Analogue I/O records have these fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EGU</td>
<td>Engineering unit string</td>
</tr>
<tr>
<td>LINR</td>
<td>Unit conversion control: No conversion, Linear, Slope, breakpoint table name</td>
</tr>
<tr>
<td>EGUL</td>
<td>Low engineering value</td>
</tr>
<tr>
<td>EGUF</td>
<td>High engineering value</td>
</tr>
<tr>
<td>ESLO</td>
<td>Unit conversion slope</td>
</tr>
<tr>
<td>EOFF</td>
<td>Unit conversion offset</td>
</tr>
</tbody>
</table>
Periodically Scanned Analog Input

- Analogue Input “Temperature”
- Reads from the Xycom XY566 ADC Card 0 Signal 0
- Gets a new value every second
- Data is converted from ADC range to 0..120 Celsius
Interrupt Scanned Binary Input

- Binary Input “VentValve”
- Reads from Allen-Bradley TTL I/O Link 0, Adaptor 0, Card 3, Signal 5
- Processed whenever value changes
- 0 = “Closed”, 1 = “Open”
- Major alarm when valve open
Most output records have these fields

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUT</td>
<td>Output link</td>
</tr>
<tr>
<td>DTYP</td>
<td>Device type</td>
</tr>
<tr>
<td>VAL</td>
<td>Engineering value</td>
</tr>
<tr>
<td>RVAL</td>
<td>Raw output value</td>
</tr>
<tr>
<td>DOL</td>
<td>Input link to fetch output value</td>
</tr>
<tr>
<td>OMSL</td>
<td>Output mode select:</td>
</tr>
<tr>
<td></td>
<td>Supervisory, Closed Loop</td>
</tr>
<tr>
<td>LOPR</td>
<td>Low operator range</td>
</tr>
<tr>
<td>HOPR</td>
<td>High operator range</td>
</tr>
</tbody>
</table>
Analogue outputs also have these fields:

OROC  Output rate of change
OIF   Incremental or Full output
OVAL  Output value
DRVH  Drive high limit
DRVL  Drive low limit
IVOA  Invalid output action
IVOV  Invalid output value
RBV   Read-back value
Passive Binary Output

- Binary Output “Solenoid”
- Controls Xycom XY220 Digital output Card 2 Signal 12
- Record is only processed by
  - Channel Access ‘put’ to a PP field (e.g. .VAL)
  - Another record writes to a PP field
  - Forward Link from another record
  - Another record reads this with PP
Break time...

5 Minute break
Links

A link is a type of field, and is one of:

- **Input link**
  - Fetches data
- **Output link**
  - Writes data
- **Forward link**
  - Points to the record to be processed once this record finishes processing
Input and Output links may be...

- Constant numeric value, e.g.:
  - 0
  - 3.1415926536
  - 1.6e-19

- Hardware link
  A hardware I/O signal selector, the format of which depends on the device support layer

- Process Variable link — the name of a record, which at run-time is resolved into
  - Database link
    Named record is in this IOC
  - Channel Access link
    Named record not found in this IOC
Hardware links

VME_IO  #Cn Sn @parm
Card, Signal

INST_IO  @parm

CAMAC_IO  #Bn Cn Nn An Fn @parm
Branch, Crate, Node, Address, Function

AB_IO  #Ln An Cn Sn @parm
Link, Adapter, Card, Signal

GPIB_IO  #Ln An @parm
Link, Address

BITBUS_IO  #Ln Nn Pn Sn @parm
Link, Node, Port, Signal

BBGPIB_IO  #Ln Bn Gn @parm
Link, Bitbus Address, GPIB Address

VXI_IO  #Vn Cn Sn @parm
or  #Vn Sn @parm
Frame, Slot, Signal
Database links

These comprise:

- The name of a record in this IOC
  
  myDb:myRecord

- An optional field name
  
  .VAL (default)

- Process Passive flag
  
  NPP (default), or PP

- Maximize Severity flag
  
  NMS (default), or MS

For example:

M1:current.RBV NPP MS

- NB: An input database link with the PP flag set that is pointing to an asynchronous input record will not wait for the new value from that record
Channel Access links

- Specified like a database link
- Name specifies a record not found in this IOC
- Use Channel Access protocol to communicate with remote IOC
- May include a field name (default .VAL)

**PP** Link flags are ignored:
- *Input links are always NPP*
- *Output links follow PP attribute of destination field*
- *This behavior is identical to all other CA clients*

**MS** Link flags apply to Input links:
- *Input links honor a given NMS (default) or MS flag*
- *Output links are always NMS*

**Additional flags for CA links:**
- **CA** Forces a “local” link to use CA
- **CP** On input link, process this record on CA monitor event
- **CPP** Like **CP** but only process if **SCAN** is Process Passive
## Link flag summary

<table>
<thead>
<tr>
<th>Type</th>
<th>Input Links</th>
<th>Output Links</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB</td>
<td>PP or NPP, MS or NMS</td>
<td>PP or NPP, MS or NMS</td>
</tr>
<tr>
<td>CA</td>
<td>Always NPP, MS or NMS, CA to force link type, CP to process this record on change, CPP is like CP but only process if SCAN=Passive</td>
<td>PP behavior of destination field, Always NMS, CA to force link type</td>
</tr>
</tbody>
</table>

Chapter 5 of the IOC Application Developer’s Guide covers record links and scanning in detail, and is worth reading.
Device Support

- Records do not access hardware directly
- The Device Support layer performs I/O operations on request
- A particular device support provides I/O for a single record type
- The DTYP field determines which device support to use
- The device support selected determines the format of the link (INP or OUT field) containing device address information
- Adding new device support does not require change to the record software
- Device support may call other software to do work for it (Driver Support)
Synchronous vs Asynchronous I/O

- EPICS rules do not allow device support to busy-wait (i.e. delay record processing while waiting for the results of a slow I/O operation)
  - Fast I/O can be handled synchronously
  - Slow operations must operate asynchronously
- Register-based VME cards usually give an immediate response: synchronous
- When called, synchronous device support performs all I/O before returning
- Serial and field-bus I/O takes a long time (>10ms) to return data: asynchronous
- Asynchronous device support starts an I/O operation when the record calls it, flagging it as incomplete by setting `PACT` to true before returning
- Once results are available (CPU interrupt), the device support calls the record’s `process()` routine to finish the record processing operations
Soft Device Support

- “Hard” Input and Output records perform hardware I/O via device support
- “Soft” records access data from other records via DB or CA links
- 2 or 3 kinds of support are provided in recent R3.14 releases:
  - Soft Channel
    - Get/Put $VAL$ through link, no units conversion preformed
  - Async Soft Channel (new, for output records only)
    - Put $VAL$ through CA link, no conversions, wait for completion
  - Raw Soft Channel
    - Inputs
      - Get $RVAL$ via input link
      - Convert $RVAL$ to $VAL$ (record-type specific)
    - Outputs
      - Convert $VAL$ to $RVAL$ (record-type specific)
      - Put $RVAL$ to output link
**Forward links**

- Usually a Database link, referring to a record in same IOC
- No flags (PP, MS etc.), although VDCT includes them erroneously
- Destination record is only processed if its SCAN field is Passive
- Does not pass a value, just causes subsequent processing
- Forward linking to another IOC via Channel Access is possible, but the link must explicitly name the PROC field of the remote record
  - In this case, the remote record does not have to be SCAN Passive
Processing chains
Which record is never processed?
How often is Input_1 processed?
The PACT field

- Every record has a boolean run-time field called **PACT** (Process Active)
- **PACT** breaks loops of linked records
- It is set to true early in the act of processing the record (but it's not the first thing that the process routine does)
  - **PACT** is true whenever a link in that record is used to get/put a value
- **PACT** is set to false after record I/O and forward link processing are finished
- A **PP** link can never make a record process if it has **PACT** true
  - Input links take the current value
  - Output links just put their value
What happens here?
Preventing records from processing

- It is useful to be able to stop an individual record from processing on some condition
- Before record-specific processing is called, a value is read through the SDIS input link into DISA (which defaults to 0 if the link is not set)
- If DISA=DISV, the record will not be processed
- The default value of the DISV field is 1
- A disabled record may be put into an alarm state by giving the desired severity in the DISS field
- The FLNK of a disabled record is never triggered
Break time...

5 Minute break
1. Every 0.1 seconds, iocCore will attempt to process the Output_1 record
2. The Output_1.PACT field is currently False, so the record is quiescent and can be processed
3. If set, the Output_1.SDIS link would be read into Output_1.DISA
4. Since DISA≠DISV, the ao record type's process() routine is called
Order of Operations (Synchronous I/O)

5. The ao's process() routine checks the `Output_1.OMSL` field; it is `closed_loop`, so
6. It sets `Output_1.PACT` to True, then
7. Reads a value through the `Output_1.DOL` link
8. The `Output_1.DOL` link contains `Calculation_1.VAL` PP so this first attempts to process the `Calculation_1` record
9. The Calculation_1.SCAN field is Passive and Calculation_1.PACT is False, so processing is possible

10. If set, the Calculation_1.SDIS link would be read into DISA

11. Since DISA ≠ DISV, the calc record type's process() routine is called
Order of Operations (Synchronous I/O)

12. The calc's process() routine sets Calculation_1.PACT to True, then
13. Starts a loop to read values from the links INPA through INPL
14. The Calculation_1.INPA link is set to Input_1.VAL PP so this first attempts to process the Input_1 record
15. The Input_1.SCAN field is Passive and Input_1.PACT is False, so processing is possible

16. If set, the Input_1.SDIS link is read into the Input_1.DISA field

17. Since DISA ≠ DISV, the ai record type's process() routine is called

18. The ai process() calls the associated device support to read a value from the hardware it's attached to
Order of Operations (Synchronous I/O)

19. The device support is synchronous, so it puts the hardware input value into the `Input_1.RVAL` field and returns to the ai record's process() code.

20. The `Input_1.PACT` field is set to True.

21. The record's timestamp field `Input_1.TIME` is set to the current time.

22. The raw value in `Input_1.RVAL` is converted to engineering units, smoothed, and the result put into the `Input_1.VAL` field.
23. The Input_1.VAL is checked against alarm limits and monitor dead-bands, and appropriate actions is taken if these are exceeded.

24. If the Forward Link field Input_1.FLNK is set, an attempt is made to process the record it points to.

25. The Input_1.PACT field is set to False, and the process() routine returns control to the Calculation_1 record.
26. The value read through the `Calculation_1.INPA` link is copied into the `Calculation_1.A` field.

27. The Calculation record type's process() routine continues to loop, reading its input links.

28. In this example only the `INPA` link is set, so the routine finishes the loop and evaluates the `Calculation_1.CALC` expression (not shown).

29. The result of the expression is put in the `Calculation_1.VAL` field.
Order of Operations (Synchronous I/O)

30. The record's timestamp field `Calculation_1.TIME` is set to the current time

31. `Calculation_1.VAL` is checked against alarm limits and monitor dead-bands, and appropriate action is taken if these are exceeded

32. If the Forward Link field `Calculation_1.FLNK` is set, an attempt is made to process the record it points to

33. The `Calculation_1.PACT` field is set to False, and the process() routine returns control to the `Output_1` record
34. The value read through the Output_1.DOL link would now be forced into the range DRVL..DRVH if those fields were set, but they aren't so it's copied to the Output_1.VAL field unchanged.

35. The Output_1.VAL value is converted from engineering to raw units and placed in Output_1.RVAL.

36. Output_1.VAL is checked against alarm limits and monitor dead-bands, and appropriate action is taken if these are exceeded.

37. The associated device support is called to write the value to the hardware.
38. The device support is synchronous, so it outputs the value to the attached hardware and returns

39. The record's timestamp field $\text{Output}_1.\text{TIME}$ is set to the current time

40. If the Forward Link field $\text{Output}_1.\text{FLNK}$ is set, an attempt is made to process the record it points to

41. The $\text{Output}_1.\text{PACT}$ field is set to False, and the process() routine returns
How are records given CPU time?

Several IOC tasks are used:
- callback (3 priorities) — I/O Interrupt
- scanEvent — Soft Event
- scanPeriod — Periodic
  - A separate task is used for each scan period
  - Faster scan rates are given a higher task priority (if supported by the IOC’s Operating System)
- Channel Access tasks use lower priority than record processing
  - If a CPU spends all its time doing I/O and record processing, you may be unable to control or monitor the IOC via the network
What could go wrong here?
Lock-sets

- Prevent a record from being processed simultaneously from two scan tasks
- A lock-set is a group of records interconnected by database links:
  - Output links
  - Forward links
  - Input links which are PP or MS
  - Any link transporting an Array
- Lock-sets are determined automatically by the IOC at start-up, or whenever a database link is added, deleted or modified

You can split a lock set with
- Channel Access links, using CA flag
- Database links which are both NPP and NMS
Alarms

- Every record has the fields
  - **SEVR**  Alarm Severity
    - *NONE, MINOR, MAJOR, INVALID*
  - **STAT**  Alarm Status (reason)
    - *READ, WRITE, UDF, HIGH, LOW, STATE, COS, CALC, DISABLE, etc.*
- Most numeric records check **VAL** against **HIHI, HIGH, LOW and LOLO** fields after the value has been determined
- The **HYST** field prevents alarm chattering
- A separate severity can be set for each numeric limit (**HHSV, HSV, LSV, LLSV**)
- Discrete (binary) records can raise alarms on entering a particular state, or on a change of state (**COS**)
Change Notification: Monitor Dead-bands

- Channel Access notifies clients that are monitoring a numeric record when
  - $\text{VAL}$ changes by more than the value in field:
    - $\text{MDEL}$ Value monitors
    - $\text{ADEL}$ Archive monitors
  - Record’s Alarm Status changes
    - $\text{HYST}$ Alarm hysteresis

- The Analogue Input record provides a smoothing filter to reduce noise on the input signal ($\text{SMOO}$)
Breakpoint Tables

- Analogue Input and Output records can do non-linear conversions from/to the raw hardware value
- Breakpoint tables interpolate values from a given table
- To use, set the record’s LINR field to the name of the breakpoint table you want to use
- Example breakpoint table (in some loaded .dbd file)

```plaintext
breaktable(typeKdegC) {
    0.000000   0.000000
    299.268700  74.000000
    660.752744 163.000000
    1104.793671 274.000000
    1702.338802 418.000000
    2902.787322 703.000000
    3427.599045 831.000000
    ...
}
```
Simulation

- Input and output record types often allow simulation of hardware interfaces
  - **SIML**  Simulation mode link
  - **SIMM**  Simulation mode value
  - **SIOL**  Simulation input link
  - **SIMS**  Simulation alarm severity
- Before using its device support, a record reads **SIMM** through the **SIML** link
- If **SIMM**=**YES**, device support is ignored; record I/O uses the **SIOL** link instead
- An alarm severity can be set whenever simulating, given by **SIMS** field
Access Security

- A networked control system must have the ability to enforce security rules
  - Who can do what from where, and when?
- In EPICS, security is enforced by the CA server (typically the IOC).
- A record is placed in the Access Security Group named in its ASG field
  - DEFAULT is used if no group name is given
- Rules for each group determine whether a CA client can read or write to records in the group, based on
  - Client user ID
  - Client IP address
  - Access Security Level of the field addressed
  - Values read from the database
Access Security Configuration File

- Security rules are loaded from an Access Security Configuration File, for example:
  
  UAG(users) {user1, user2}
  HAG(hosts) {host1, host2}
  ASG(DEFAULT) {
      RULE(1, READ)
      RULE(1, WRITE) {
          UAG(users)
          HAG(hosts)
      }
  }

- If no security file is loaded, Security will be turned off and nothing refused
- For more details and the rule syntax, see Chapter 8 of the IOC Application Developers Guide