# EPICS:
# An Introduction

# What is EPICS

- A collaboration of the controls groups of many research organizations that use the EPICS tool-kit.
- A distributed architecture that supports a wide range of solutions from small test stands to large integrated facilities.
- A set of tools that reduces software application and maintenance costs by providing:

  Configuration tools in place of programming

  A large installed base of tested software

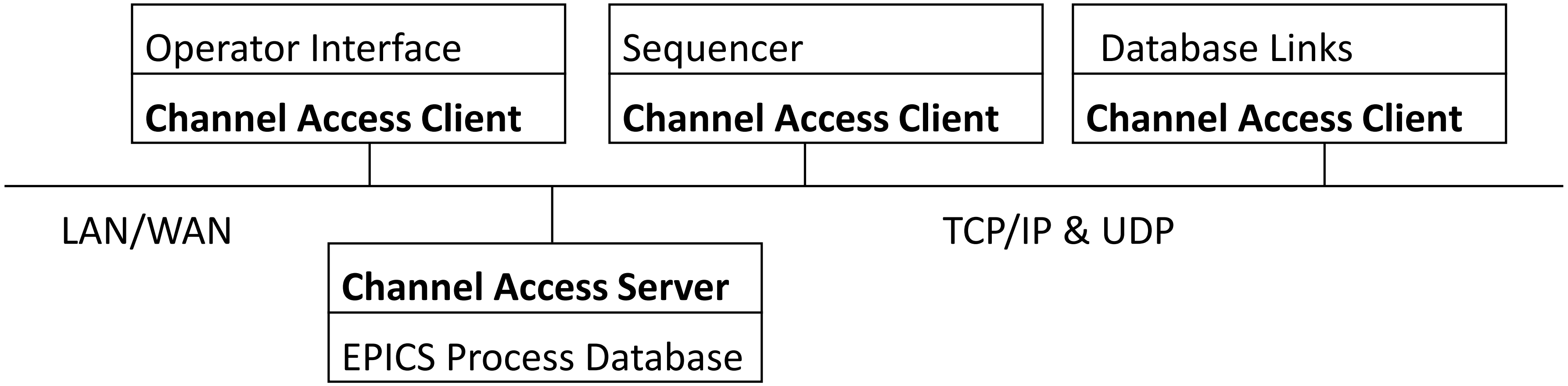
  A modular design that supports incremental upgrades

  Well defined interfaces for extensions at every level

# EPICS architecture

- EPICS software architecture is client/server based - with independent data stores providing read/write access directly between any two points
- EPICS V3 is physically a flat architecture of front-end controllers and operator workstations that communicate via TCP/IP and UDP
  - System scales through the addition of new computers
  - Physical hierarchy is made through bridges, routers, or a gateway
  - Network bandwidth is the primary limiting factor

# EPICS architecture


Operator Interface
**Channel Access Client**
Sequencer
**Channel Access Client**
Database Links
**Channel Access Client**
LAN/WAN
TCP/IP & UDP
**Channel Access Server**
EPICS Process Database

- Server: Provides read/write connections to information in this node to any client on the network through channel access client calls. The data resides here!
- Client: Provides read/write connections to any subsystem on the network with a channel access server
- Services: Dynamic Channel Location, Get, Put, Monitor, Access Control, Connection Monitoring, Automatic Reconnect, Conversion to client types, Composite Data Structures

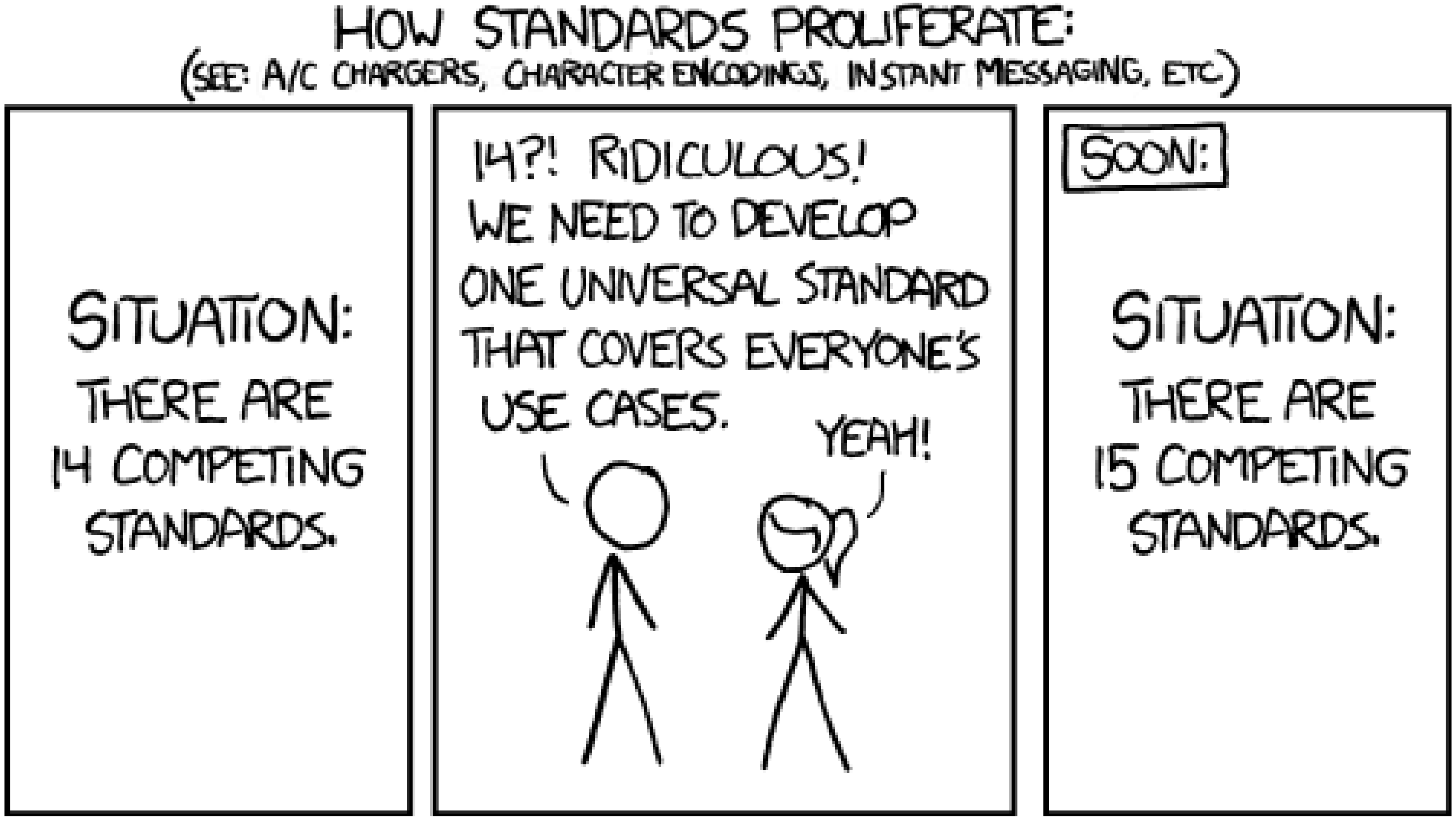
# Terminology

- EPICS has network servers (IOCs) and clients (OPIs and IOCs)
- **IOC** = Input Output Controller
  The server application
- **OPI** = Operator Interface Generic term for client application
- **CA** = Channel Access
  The network protocol (analogous to HTTP)
- **PV** = Process Variable
  The unit/quantum of addressable data with CA protocol

# What is a PV?

- A name identifying some signal value
  - In hardware
    A temperature or pressure reading
  - In software
    connection status or error counter
- PV Name examples
  - **UT:BR-Cu:2{Pmp:1}PD-I**
    - *Booster cooling water skid, copper system #2, pump #1, differential pressure indicator.*
  - **SR:C09-PS:RGB1{PS:CXM1B-ASM:XG-CH1}T:1-I**
    - *SR cell #9 rack group B1 corrector power supply 1B heat exchanger, channel #1 temperature indicator (chain #1)*

# PV Naming Standards


HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC.)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

https://xkcd.com/927/

# Basic CA Client Tools

- Always available
- Simple and good for debugging, but not much more

- caput – Change a setting
- caget – Fetch the present value once
- camonitor – Watch the value until interrupted
- cainfo – Fetch diagnostic info

# Basic CA Client Tools (2)

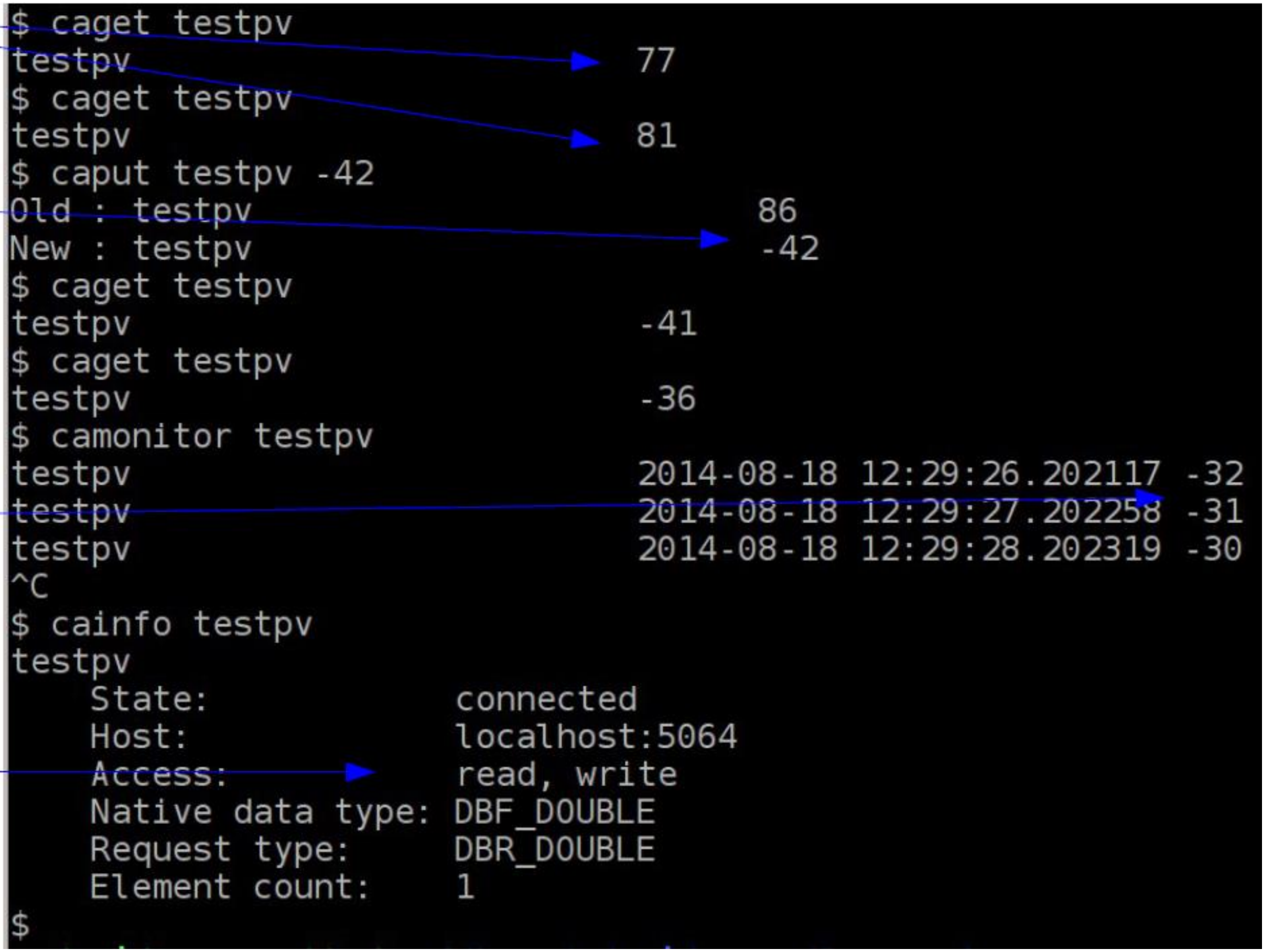
Fetch value

It changes!

Write a new value

See how it changes
In time

Useful information
About this PV



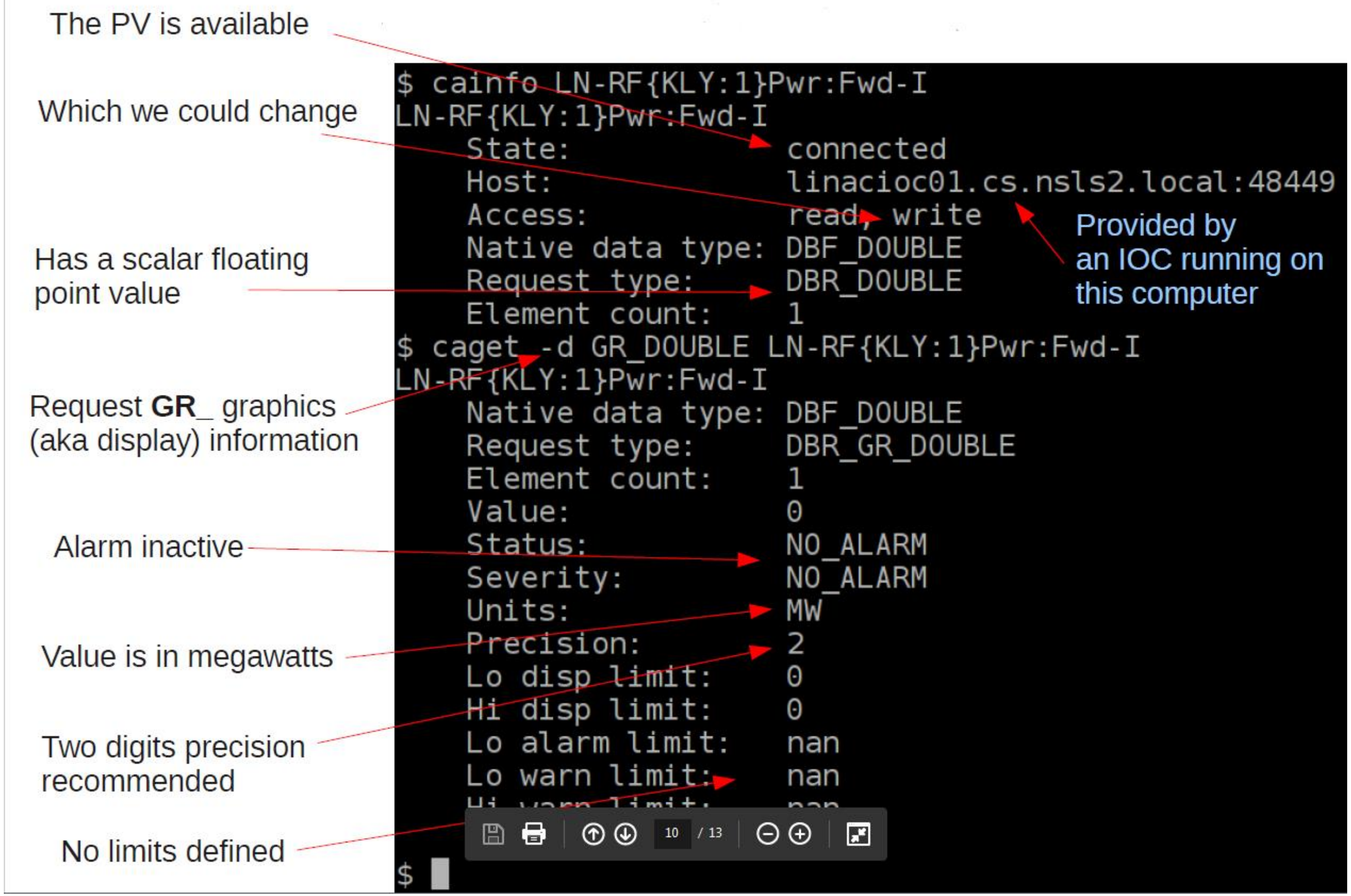
```
$ caget testpv
testpv                         77
$ caget testpv
testpv                         81
$ caput testpv -42
Old : testpv                         86
New : testpv                         -42
$ caget testpv
testpv                         -41
$ caget testpv
testpv                         -36
$ camonitor testpv
testpv                         2014-08-18 12:29:26.202117 -32
testpv                         2014-08-18 12:29:27.202258 -31
testpv                         2014-08-18 12:29:28.202319 -30
^C
$ cainfo testpv
testpv
    State:            connected
    Host:             localhost:5064
    Access:           read, write
    Native data type: DBF_DOUBLE
    Request type:     DBR_DOUBLE
    Element count:    1
$
```

# What comes with a PV?

- **Value**: Integer, Float, String
- Absolute **time** of last change
- **Alarm** state (severity and status codes)
- **Limits** (alarm, display, and control)
- **Units** string
- **Precision** (number of decimal digits)
- List of **states** strings (for enumeration)

  eg. ['invalid','moving','closed','open']

# Investigating a PV



```
$ cainfo LN-RF{KLY:1}Pwr:Fwd-I
LN-RF{KLY:1}Pwr:Fwd-I
    State:            connected
    Host:             linacioc01.cs.nsls2.local:48449
    Access:           read, write
    Native data type: DBF_DOUBLE
    Request type:     DBR_DOUBLE
    Element count:    1
$ caget -d GR_DOUBLE LN-RF{KLY:1}Pwr:Fwd-I
LN-RF{KLY:1}Pwr:Fwd-I
    Native data type: DBF_DOUBLE
    Request type:     DBR_GR_DOUBLE
    Element count:    1
    Value:            0
    Status:           NO_ALARM
    Severity:         NO_ALARM
    Units:            MW
    Precision:        2
    Lo disp limit:    0
    Hi disp limit:    0
    Lo alarm limit:   nan
    Lo warn limit:    nan
    Hi warn limit:    nan
$
```

The PV is available

Which we could change

Has a scalar floating point value

Request **GR_** graphics (aka display) information

Alarm inactive

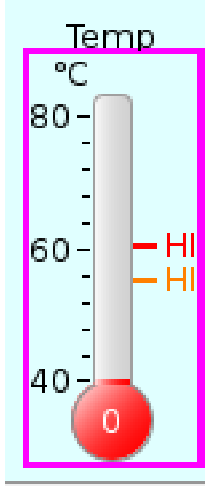
Value is in megawatts

Two digits precision recommended

No limits defined

Provided by an IOC running on this computer

# Investigating a PV (2)



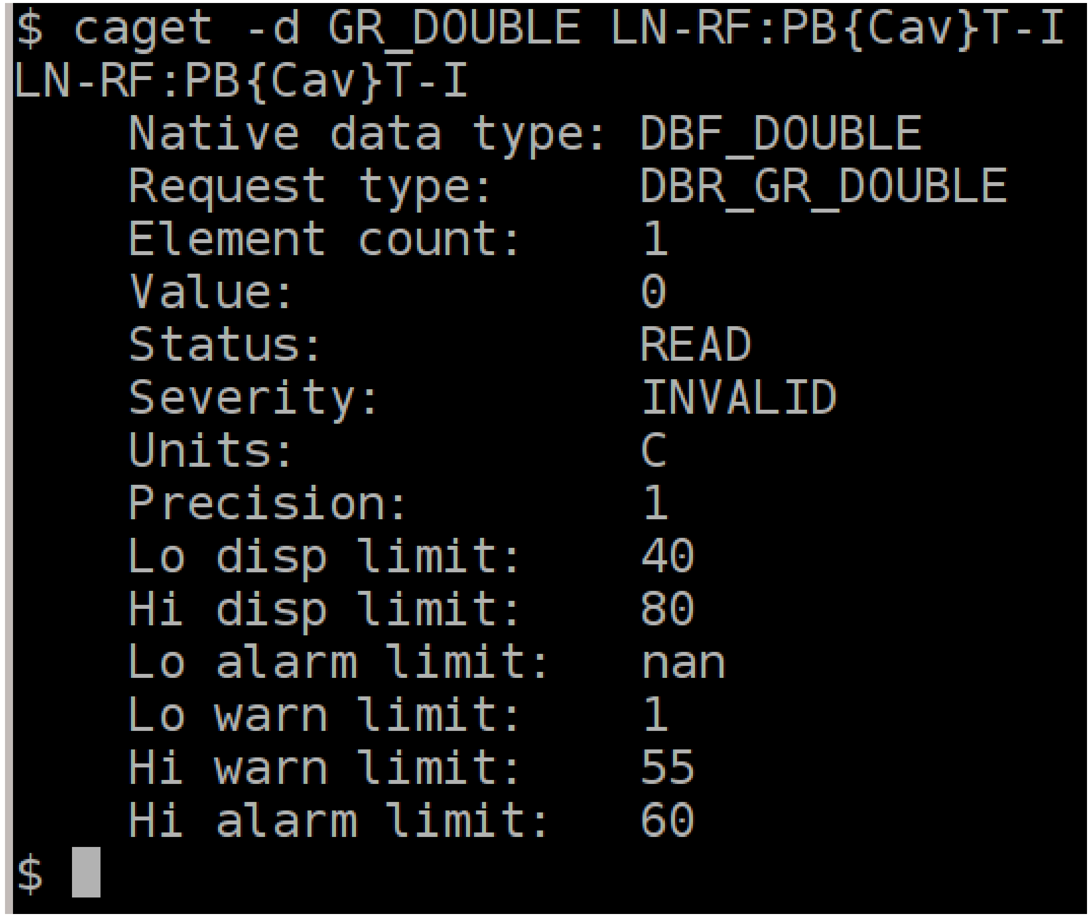
Something is wrong here!

Alarm ranges are defined

Display range: 40 to 80 C

MINOR range: 1 to 55 C

MAJOR range: < 60 C



```
$ caget -d GR_DOUBLE LN-RF:PB{Cav}T-I
LN-RF:PB{Cav}T-I
    Native data type: DBF_DOUBLE
    Request type:     DBR_GR_DOUBLE
    Element count:    1
    Value:            0
    Status:           READ
    Severity:         INVALID
    Units:            C
    Precision:        1
    Lo disp limit:    40
    Hi disp limit:    80
    Lo alarm limit:   nan
    Lo warn limit:    1
    Hi warn limit:    55
    Hi alarm limit:   60
$ 
```

# Alarm State

- Severity
  - NO_ALARM (0) - Normal
  - MINOR (1) – Warning (yellow/orange)
  - MAJOR (2) - Error condition (red)
  - INVALID (3) - Value not meaningful (white/violet)
    - eg. device is powered off or disconnected.
- Status
  - READ, WRITE, …
  - Knowing status codes isn't as important

# Understanding Alarms

- Alarms are subjective
  - Not all MAJOR alarms are equal
- OPI clients can highlight alarming PVs with a colored border
- Specialized Alarm clients (ALH or Beast) which aggregate large number of alarms.