whatrecord

Ken Lauer
LCLS, SLAC National Accelerator Laboratory
September 23rd, 2022

https://github.com/pcdshub/whatrecord/

Motivation

EPICS IOCs/modules/extensions are comprised of a conglomeration of weird file formats:

- Process database files (.db)
- Database definition files (.dbd)
- Template / substitutions files
- IOC shell scripts (st.cmd)
- StreamDevice protocols (.proto)
- State notation language programs (.st)
- Gateway configuration (.pvlist)
- Access security files (.acf)
- Facility-specific things like LCLS’s IOC manager configuration
- Build system Makefiles
- …

At the LCLS, we have somewhere around 3,000 IOC instances in total – including those from the accelerator side and the photon side.

Motivation

Wouldn’t it be neat if…

- We could easily parse those formats outside of an IOC and represent them in a widely-used interchange format like JSON?
- We could understand a bit better what’s in our existing IOCs, whether they are deployed and running or not?
- We could see how different records, different IOCs, all relate to one another… without even running an IOC?
- We could somehow jump from a PV name to its database file/record definition/st.cmd/IOC?
- We could dive a bit deeper by linking records to PLC code? To StreamDevice protocol information? To gateway access rules? Shell commands to source code, even?

whatrecord: supported parsing tools

Parse any of the following into intuitive Python dataclasses using lark:

- Database files (V3 or V4/V7), database definitions, template/substitution files
- Access security configuration files
- Autosave .sav files
- Gateway pvlist configuration files
- StreamDevice protocol files
- snlseq/sequencer state machine parsing

Interpret IOC shell scripts (i.e., st.cmd) and track:

- What files were loaded during startup?
- What records are available?
- What errors were found?
- What file and line did record X get loaded?
- Inter- or intra-IOC record relationships

https://github.com/pcdshub/whatrecord https://pypi.org/project/whatrecord/

whatrecord core: other things

EPICS build system `Makefile` introspection

- *sumo*-inspired implementation, but only JSON details or dependency graph output

GDB Python script that inspects binary symbols to find IOC shell commands, variables and source code context

```
dbLoadRecords [str: filename] [str: substitutions]
    modules/database/src/ioc/db/dbIocRegister.c line 53
```

Accurate EPICS macro handling (epics-base macLib, wrapped with Cython in `epicsmacrolib` (GitHub; PyPI)

Plugins for loading happi devices, TwinCAT PLC projects, IOC information from LCLS's IOC manager, …

- Process database record -> PLC source code definition

whatrecord: accessing the parsed information

- Python API, command-line tools for some of the above things
- And a web-based API/backend server to monitor IOC scripts and serve IOC/record information.
  - Load up all EPICS IOCs (either user-specified or those listed in LCLS’s IOC manager tool)
    - Load the startup scripts
    - Load all the databases and supported files
  - Monitor loaded files for changes
  - Provide a backend service for querying the information

- Based on the backend server, provide a frontend for easy access to that information
  - Vue.js-based frontend single-page application
  - Search for records/IOCs/etc by name and dig into the details...

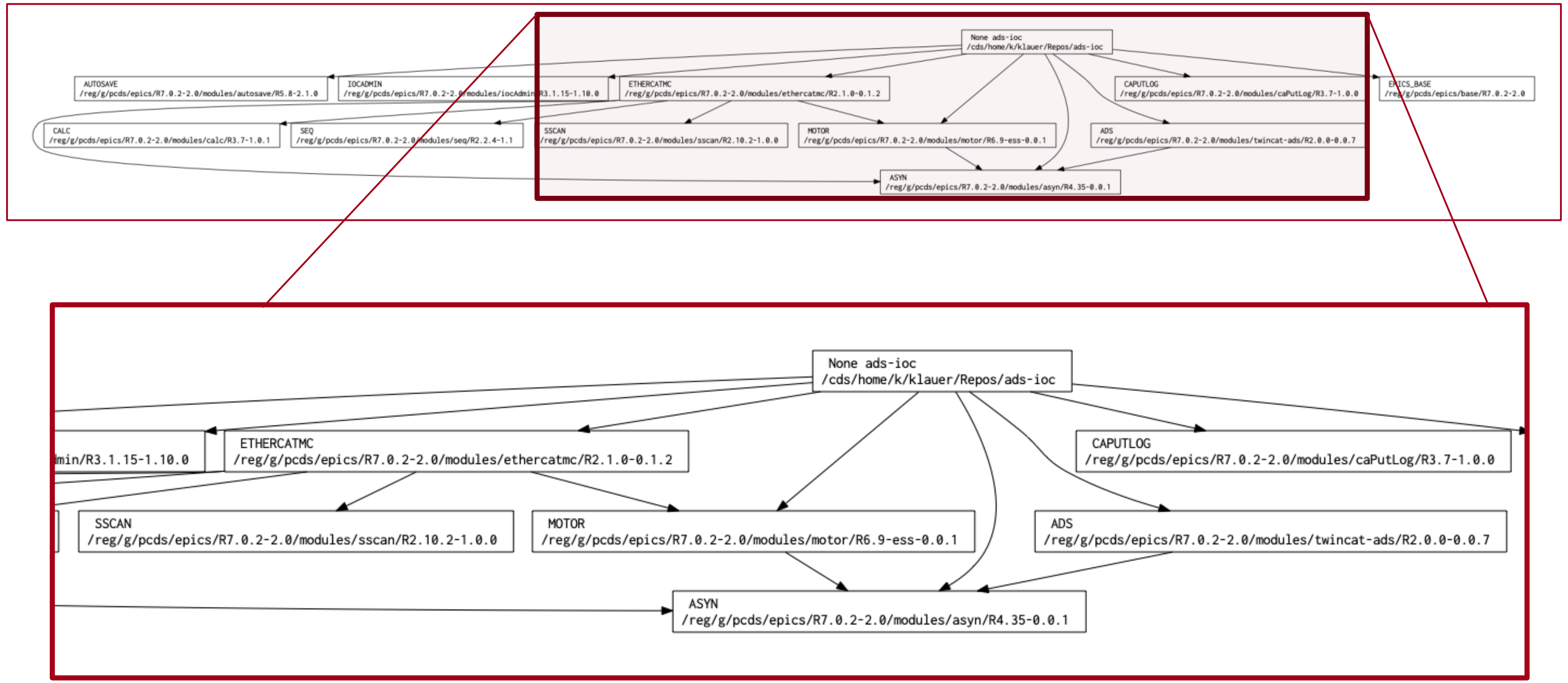
~~Demo~~ Some screenshots

`whatrecord parse`: Quick example with jq

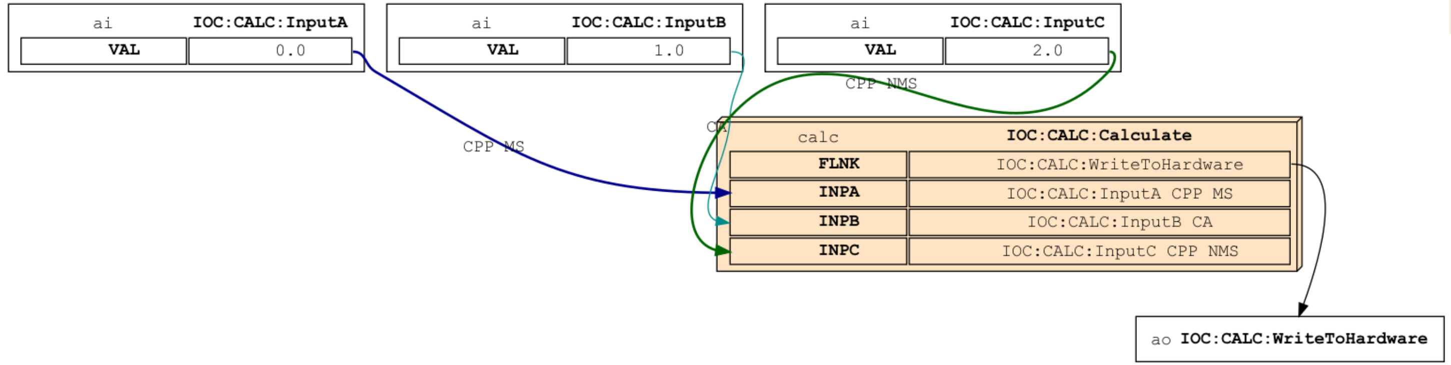
```
$ whatrecord parse whatrecord/tests/iocs/db/pva/iq.db |
    jq '.records[] | [.name, .record_type, .fields.OUT.value]'
[
  "$(PREFIX)Rate",
  "ao",
  "$(PREFIX)dly_.ODLY NPP"
]
[
  "$(PREFIX)Delta",
  "ao",
  null
]
...
```

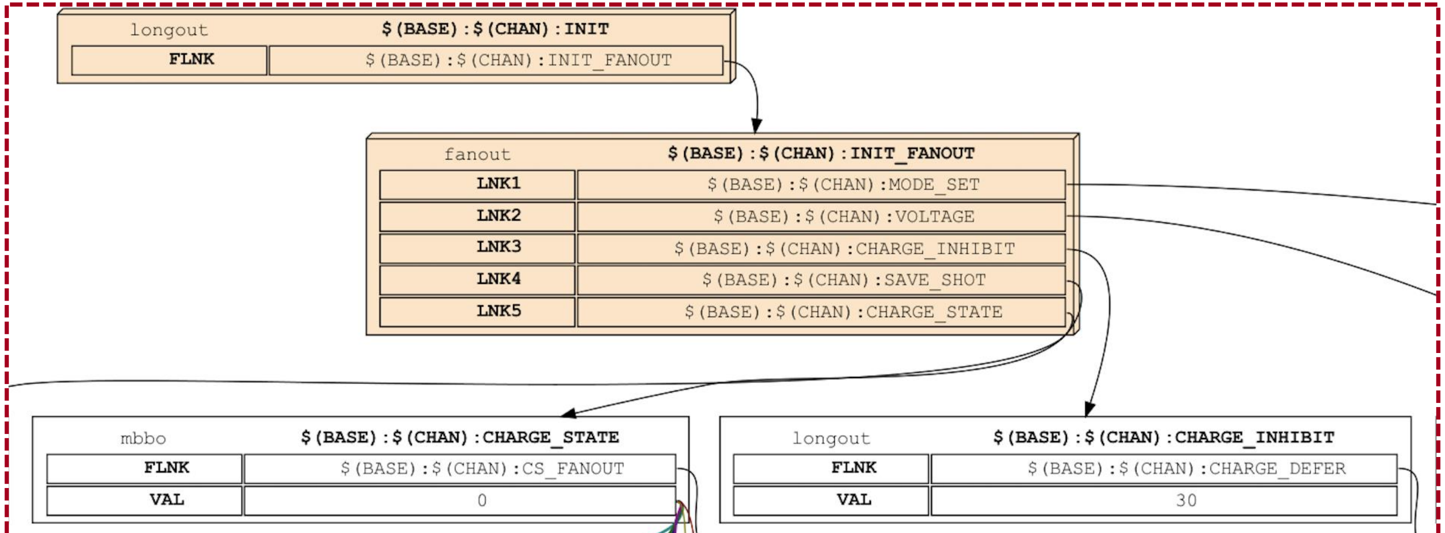
```
$ whatrecord parse whatrecord/tests/iocs/db/pva/iq.db |
    jq '.records[] | [ .name, .info["Q:group"]]'
[
  "$(PREFIX)Rate",
  null
]
[
  "$(PREFIX)Phase:I",
  {
    "$(PREFIX)iq": {
      "phas.i": {
        "+type": "plain",
        "+channel": "VAL"
      }
    }
  }
]
...
```

`whatrecord deps`: Makefile-derived dependency graph tool


None ads-ioc
/cds/home/k/klauer/Repos/ads-ioc
AUTOSAVE
/reg/g/pcds/epics/R7.0.2-2.0/modules/autosave/R5.8-2.1.0
IOCADMIN
/reg/g/pcds/epics/R7.0.2-2.0/modules/iocAdmin/R3.1.15-1.10.0
ETHERCATMC
/reg/g/pcds/epics/R7.0.2-2.0/modules/ethercatmc/R2.1.0-0.1.2
CAPUTLOG
/reg/g/pcds/epics/R7.0.2-2.0/modules/caPutLog/R3.7-1.0.0
EPICS_BASE
/reg/g/pcds/epics/base/R7.0.2-2.0
CALC
/reg/g/pcds/epics/R7.0.2-2.0/modules/calc/R3.7-1.0.1
SEQ
/reg/g/pcds/epics/R7.0.2-2.0/modules/seq/R2.2.4-1.1
SSCAN
/reg/g/pcds/epics/R7.0.2-2.0/modules/sscan/R2.10.2-1.0.0
MOTOR
/reg/g/pcds/epics/R7.0.2-2.0/modules/motor/R6.9-ess-0.0.1
ADS
/reg/g/pcds/epics/R7.0.2-2.0/modules/twincat-ads/R2.0.0-0.0.7
ASYN
/reg/g/pcds/epics/R7.0.2-2.0/modules/asyn/R4.35-0.0.1

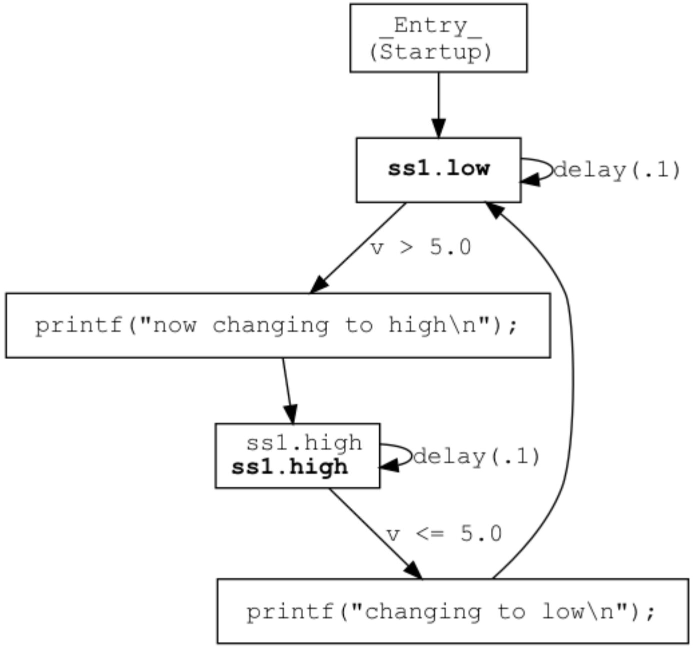
`whatrecord graph`: Intra/inter-IOC record graphs

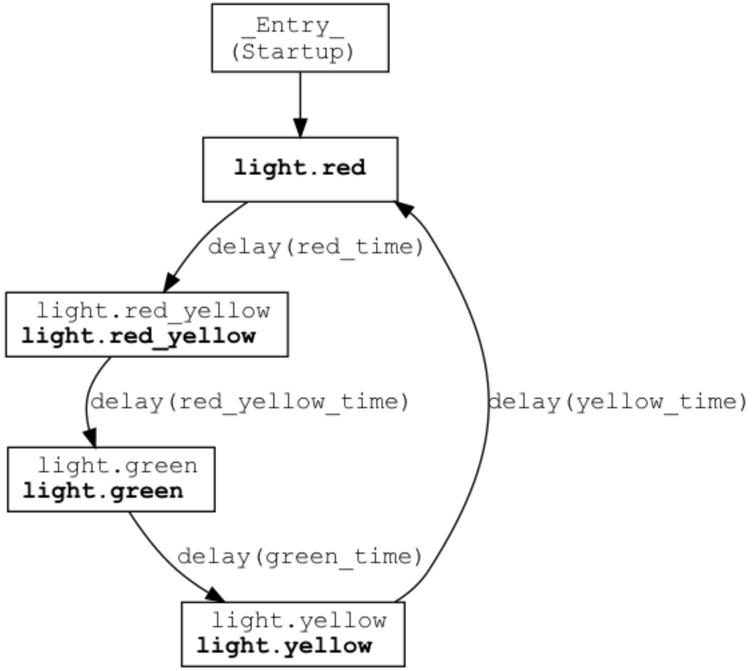




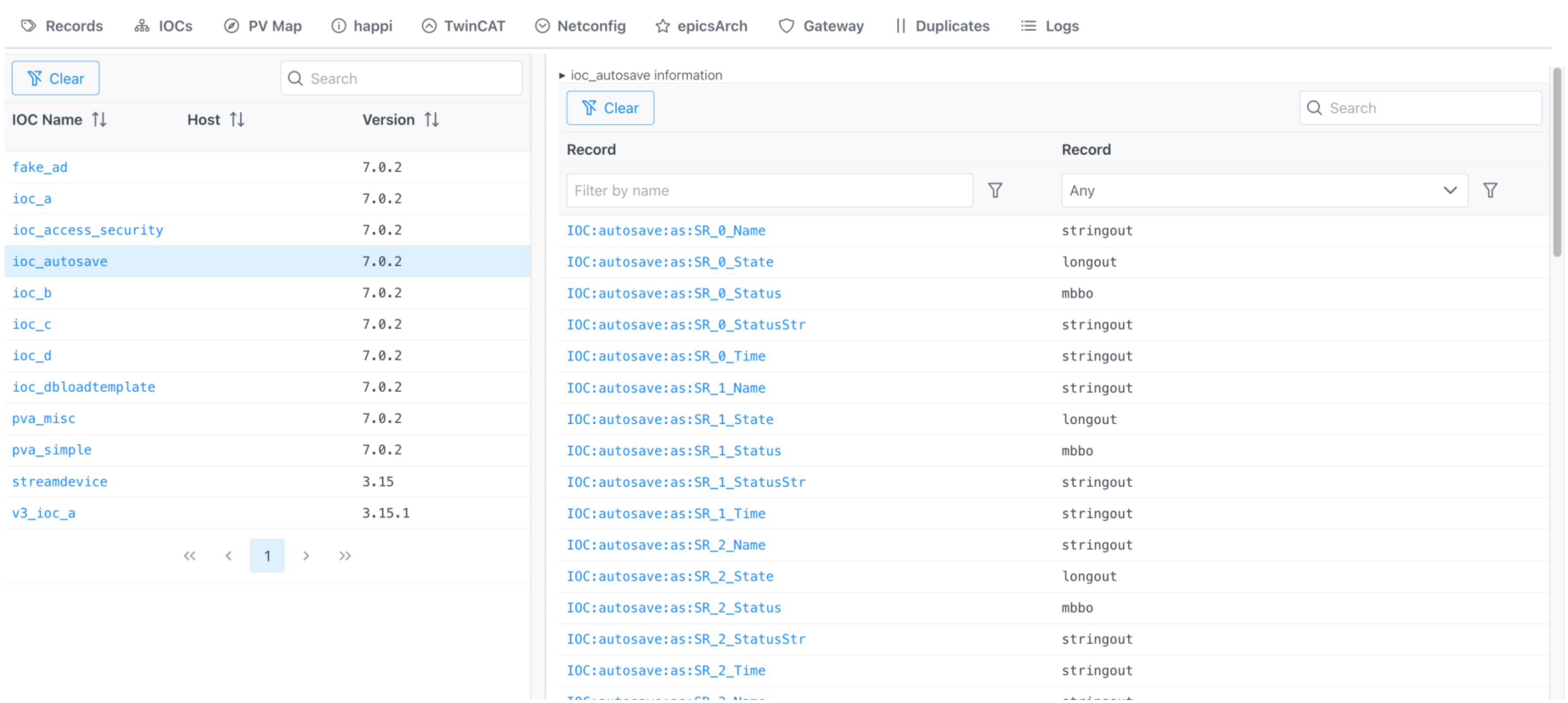
longout $(BASE):$(CHAN):INIT
FLNK $(BASE):$(CHAN):INIT_FANOUT
fanout $(BASE):$(CHAN):INIT_FANOUT
LNK1 $(BASE):$(CHAN):MODE_SET
LNK2 $(BASE):$(CHAN):VOLTAGE
LNK3 $(BASE):$(CHAN):CHARGE_INHIBIT
LNK4 $(BASE):$(CHAN):SAVE_SHOT
LNK5 $(BASE):$(CHAN):CHARGE_STATE
mbbo $(BASE):$(CHAN):CHARGE_STATE
FLNK $(BASE):$(CHAN):CS_FANOUT
VAL 0
longout $(BASE):$(CHAN):CHARGE_INHIBIT
FLNK $(BASE):$(CHAN):CHARGE_DEFER
VAL 30

`whatrecord graph`: State notation language transition diagrams





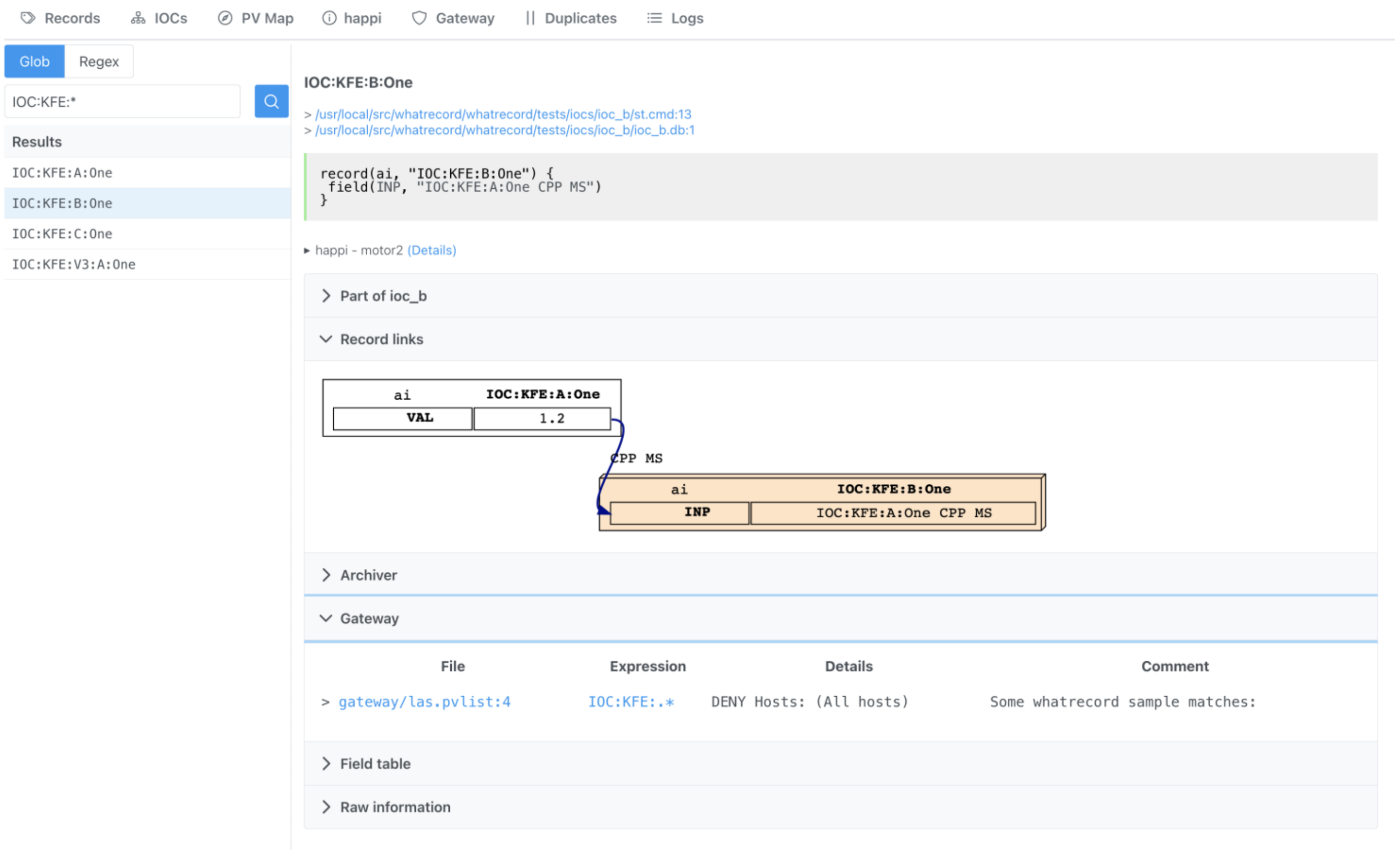
`whatrecord server` Vue.js frontend: IOC listing



IOC Name	Host	Version
fake_ad		7.0.2
ioc_a		7.0.2
ioc_access_security		7.0.2
ioc_autosave		7.0.2
ioc_b		7.0.2
ioc_c		7.0.2
ioc_d		7.0.2
ioc_dbloadtemplate		7.0.2
pva_misc		7.0.2
pva_simple		7.0.2
streamdevice		3.15
v3_ioc_a		3.15.1

Record	Record
IOC:autosave:as:SR_0_Name	stringout
IOC:autosave:as:SR_0_State	longout
IOC:autosave:as:SR_0_Status	mbbo
IOC:autosave:as:SR_0_StatusStr	stringout
IOC:autosave:as:SR_0_Time	stringout
IOC:autosave:as:SR_1_Name	stringout
IOC:autosave:as:SR_1_State	longout
IOC:autosave:as:SR_1_Status	mbbo
IOC:autosave:as:SR_1_StatusStr	stringout
IOC:autosave:as:SR_1_Time	stringout
IOC:autosave:as:SR_2_Name	stringout
IOC:autosave:as:SR_2_State	longout
IOC:autosave:as:SR_2_Status	mbbo
IOC:autosave:as:SR_2_StatusStr	stringout
IOC:autosave:as:SR_2_Time	stringout

Web frontend: record details



IOC:KFE:*

IOC:KFE:A:One
IOC:KFE:B:One
IOC:KFE:C:One
IOC:KFE:V3:A:One

IOC:KFE:B:One

> /usr/local/src/whatrecord/whatrecord/tests/iocs/ioc_b/st.cmd:13
> /usr/local/src/whatrecord/whatrecord/tests/iocs/ioc_b/ioc_b.db:1

```
record(ai, "IOC:KFE:B:One") {
 field(INP, "IOC:KFE:A:One CPP MS")
}
```

▸ happi - motor2 (Details)

> Part of ioc_b

⌄ Record links

ai	IOC:KFE:A:One
VAL	1.2

CPP MS

ai	IOC:KFE:B:One
INP	IOC:KFE:A:One CPP MS

> Archiver

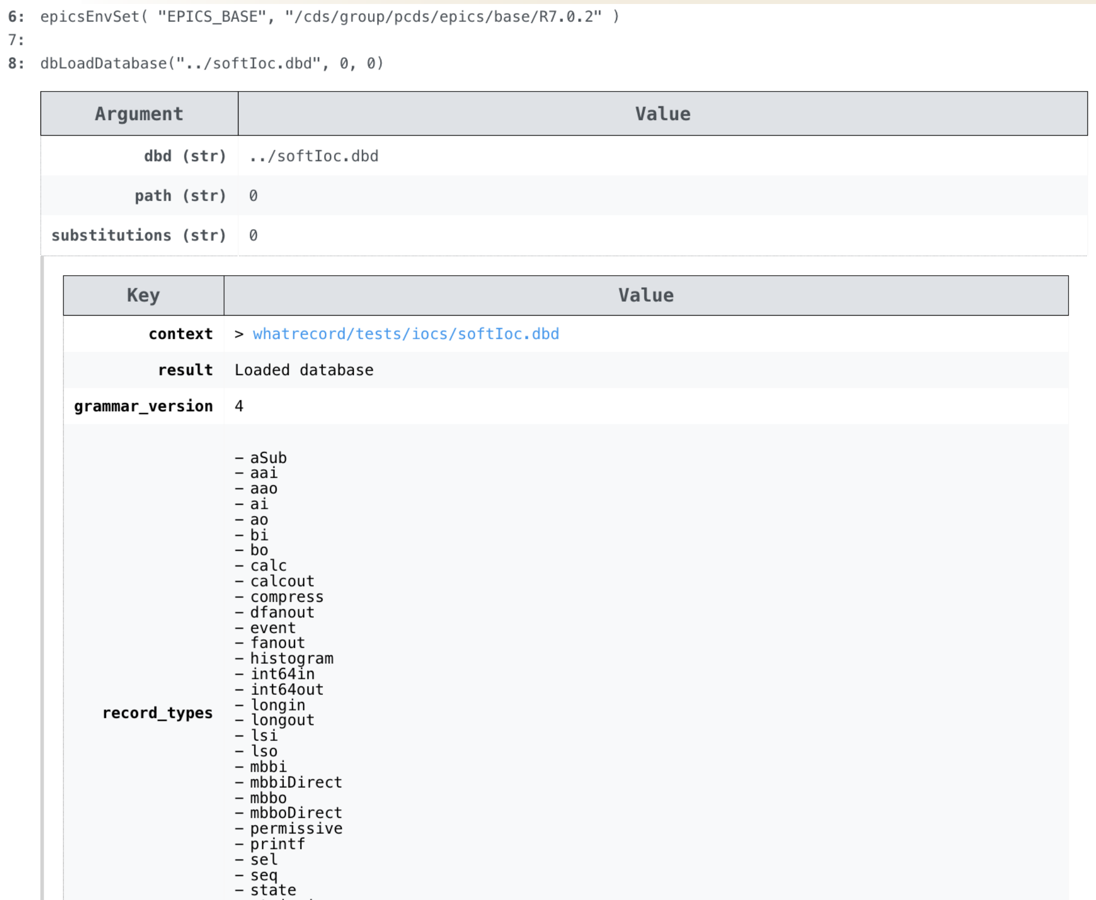
⌄ Gateway

File	Expression	Details	Comment
> gateway/las.pvlist:4	IOC:KFE:.*	DENY Hosts: (All hosts)	Some whatrecord sample matches:

> Field table

> Raw information

Web frontend: startup script line information

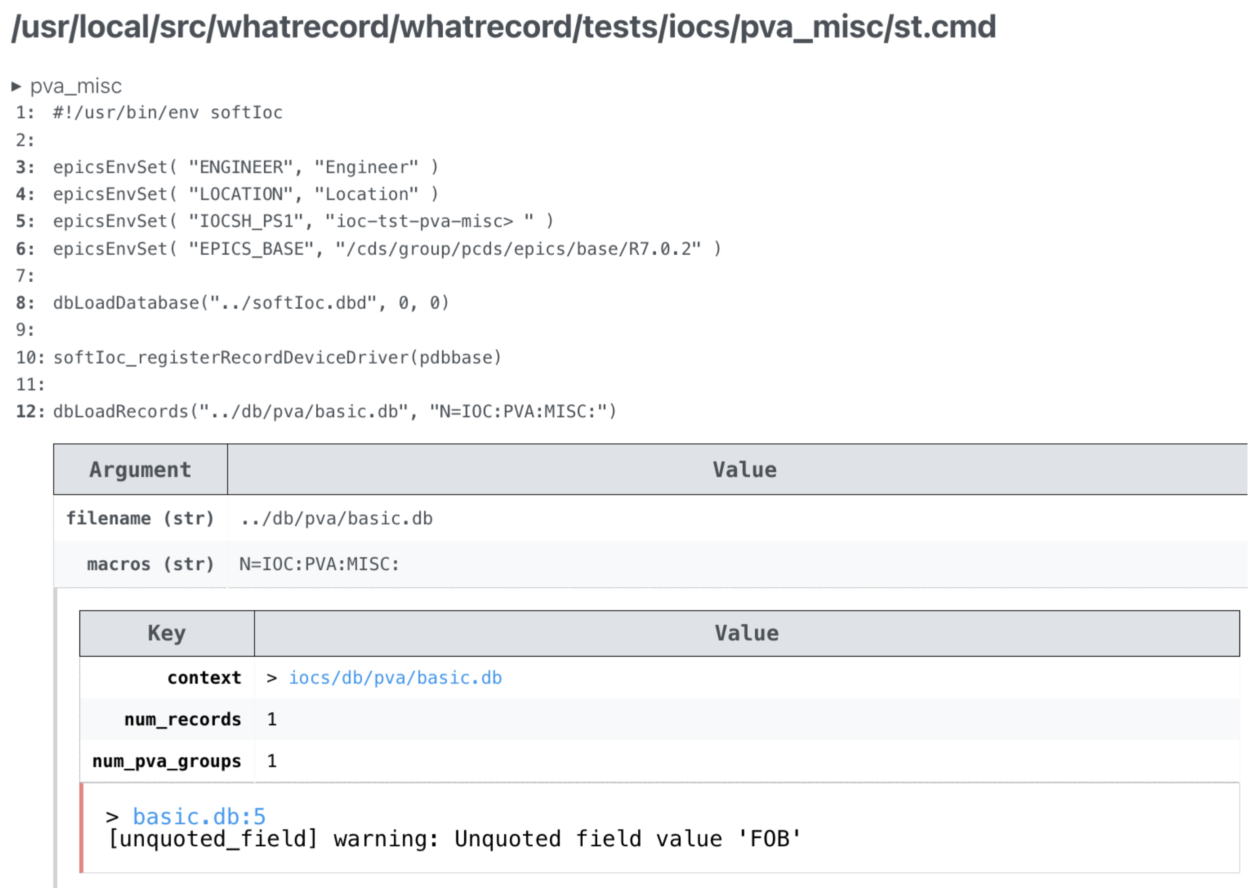


```
epicsEnvSet( "EPICS_BASE", "/cds/group/pcds/epics/base/R7.0.2" )

dbLoadDatabase("../softIoc.dbd", 0, 0)
```

Argument	Value
dbd (str)	../softIoc.dbd
path (str)	0
substitutions (str)	0

Key	Value
context	> whatrecord/tests/iocs/softIoc.dbd
result	Loaded database
grammar_version	4
record_types	– aSub – aai – aao – ai – ao – bi – bo – calc – calcout – compress – dfanout – event – fanout – histogram – int64in – int64out – longin – longout – lsi – lso – mbbi – mbbiDirect – mbbo – mbboDirect – permissive – printf – sel – seq – state

Web frontend: startup script line lint



/usr/local/src/whatrecord/whatrecord/tests/iocs/pva_misc/st.cmd

▸ pva_misc

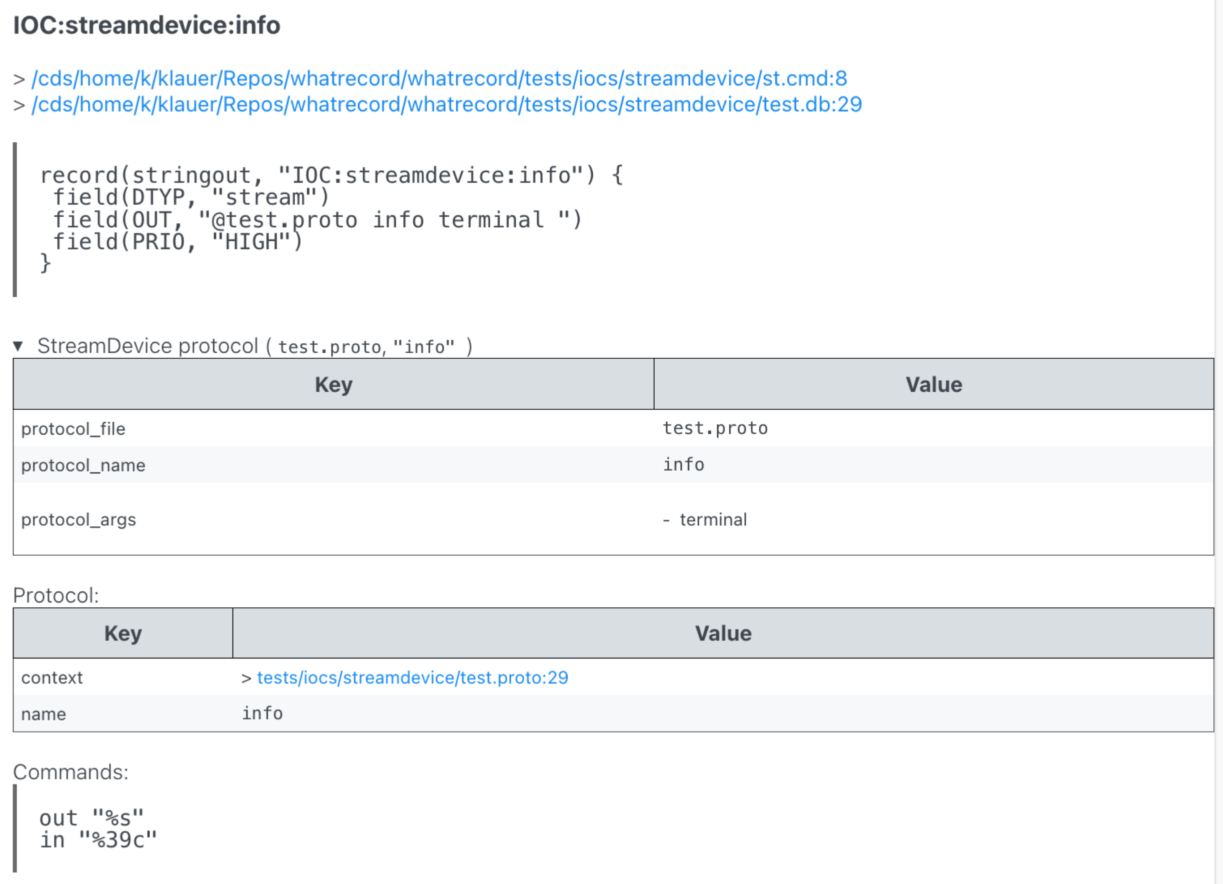
```
#!/usr/bin/env softIoc

epicsEnvSet( "ENGINEER", "Engineer" )
epicsEnvSet( "LOCATION", "Location" )
epicsEnvSet( "IOCSH_PS1", "ioc-tst-pva-misc> " )
epicsEnvSet( "EPICS_BASE", "/cds/group/pcds/epics/base/R7.0.2" )

dbLoadDatabase("../softIoc.dbd", 0, 0)

softIoc_registerRecordDeviceDriver(pdbbase)

dbLoadRecords("../db/pva/basic.db", "N=IOC:PVA:MISC:")
```

Argument	Value
filename (str)	../db/pva/basic.db
macros (str)	N=IOC:PVA:MISC:

Key	Value
context	> iocs/db/pva/basic.db
num_records	1
num_pva_groups	1

```
> basic.db:5
[unquoted_field] warning: Unquoted field value 'FOB'
```

Web frontend: Record to StreamDevice information



IOC:streamdevice:info

> /cds/home/k/klauer/Repos/whatrecord/whatrecord/tests/iocs/streamdevice/st.cmd:8
> /cds/home/k/klauer/Repos/whatrecord/whatrecord/tests/iocs/streamdevice/test.db:29

```
record(stringout, "IOC:streamdevice:info") {
 field(DTYP, "stream")
 field(OUT, "@test.proto info terminal ")
 field(PRIO, "HIGH")
}
```

▼ StreamDevice protocol (`test.proto`, `"info"`)

Key	Value
protocol_file	`test.proto`
protocol_name	`info`
protocol_args	- terminal

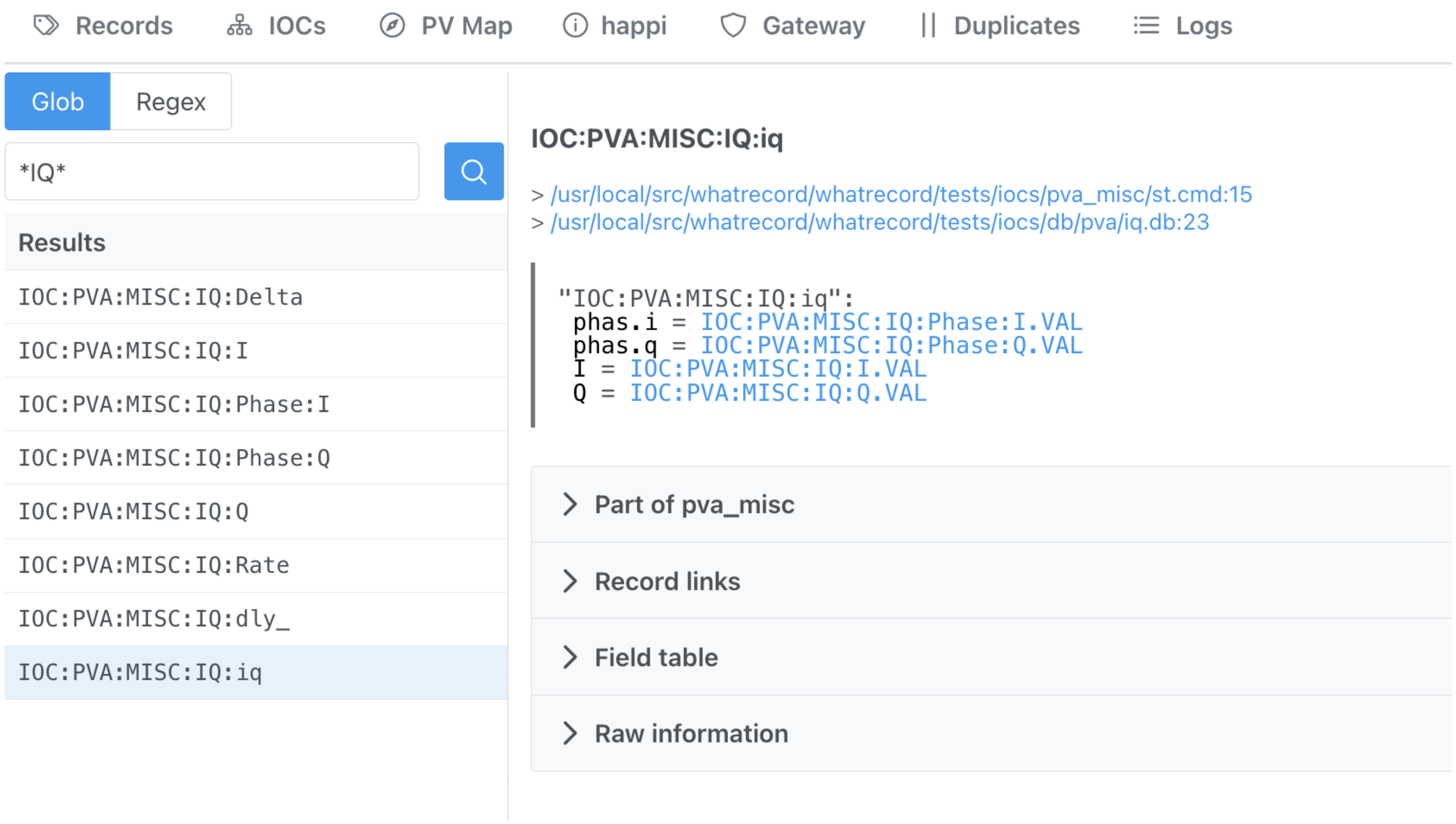
Protocol:

Key	Value
context	> tests/iocs/streamdevice/test.proto:29
name	`info`

Commands:

```
out "%s"
in "%39c"
```

Web frontend: PVAccess group information



Results

```
IOC:PVA:MISC:IQ:Delta
IOC:PVA:MISC:IQ:I
IOC:PVA:MISC:IQ:Phase:I
IOC:PVA:MISC:IQ:Phase:Q
IOC:PVA:MISC:IQ:Q
IOC:PVA:MISC:IQ:Rate
IOC:PVA:MISC:IQ:dly_
IOC:PVA:MISC:IQ:iq
```

IOC:PVA:MISC:IQ:iq

> /usr/local/src/whatrecord/whatrecord/tests/iocs/pva_misc/st.cmd:15
> /usr/local/src/whatrecord/whatrecord/tests/iocs/db/pva/iq.db:23

```
"IOC:PVA:MISC:IQ:iq":
 phas.i = IOC:PVA:MISC:IQ:Phase:I.VAL
 phas.q = IOC:PVA:MISC:IQ:Phase:Q.VAL
 I = IOC:PVA:MISC:IQ:I.VAL
 Q = IOC:PVA:MISC:IQ:Q.VAL
```

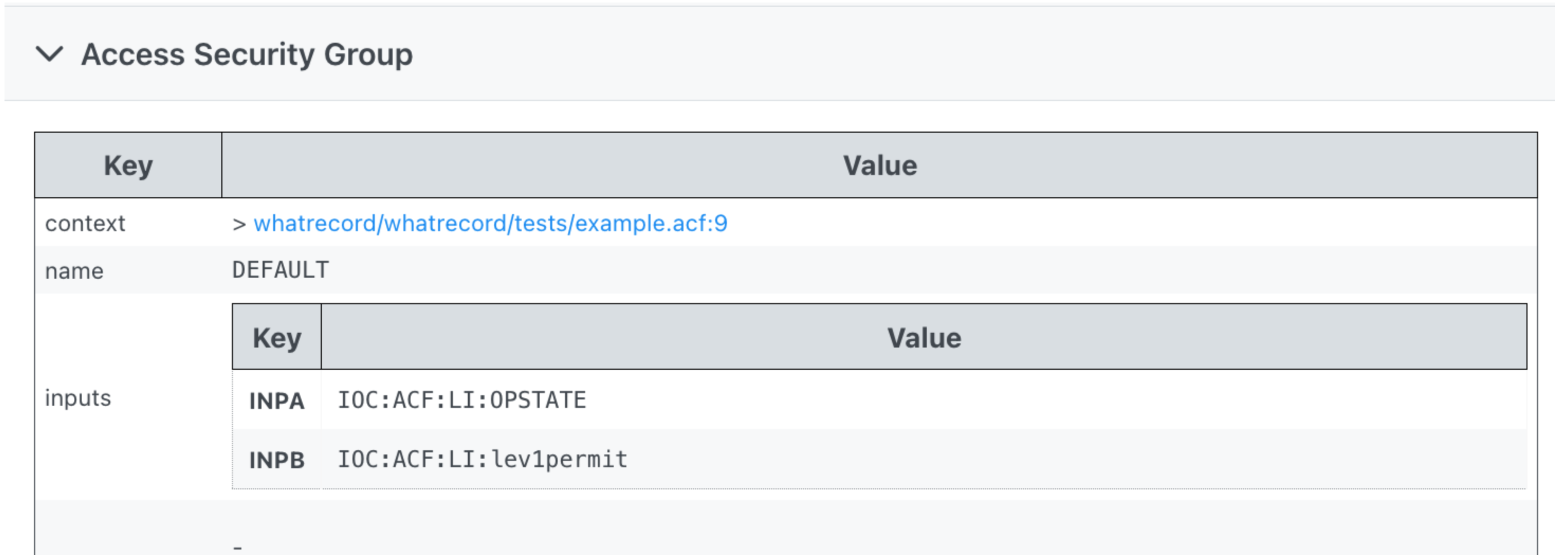
> Part of pva_misc

> Record links

> Field table

> Raw information

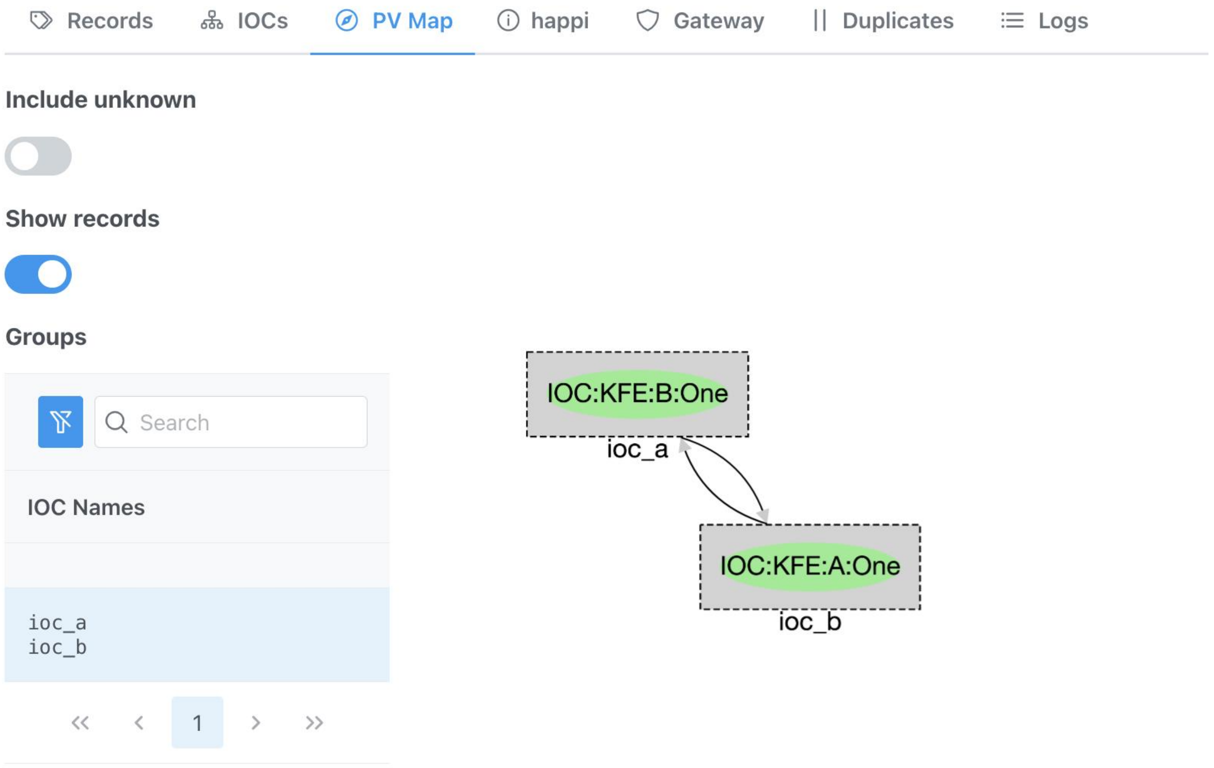
Web frontend: ASGs



Access Security Group

Key	Value
context	> whatrecord/whatrecord/tests/example.acf:9
name	DEFAULT
inputs	<table><tr><th>Key</th><th>Value</th></tr><tr><td>INPA</td><td>IOC:ACF:LI:OPSTATE</td></tr><tr><td>INPB</td><td>IOC:ACF:LI:lev1permit</td></tr></table>
	-

Web frontend: IOC/record relationship map



Include unknown

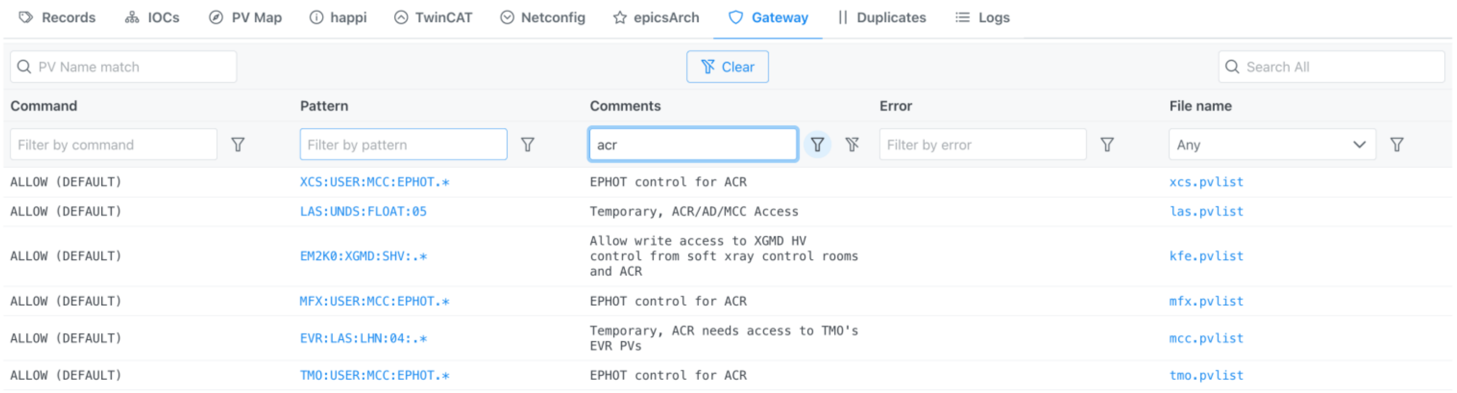
Show records

Groups

IOC Names
ioc_a ioc_b

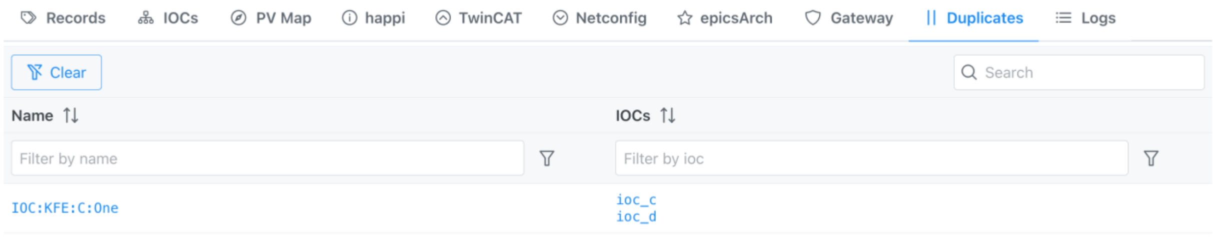
Web frontend: miscellaneous

Gateway pvlist entries



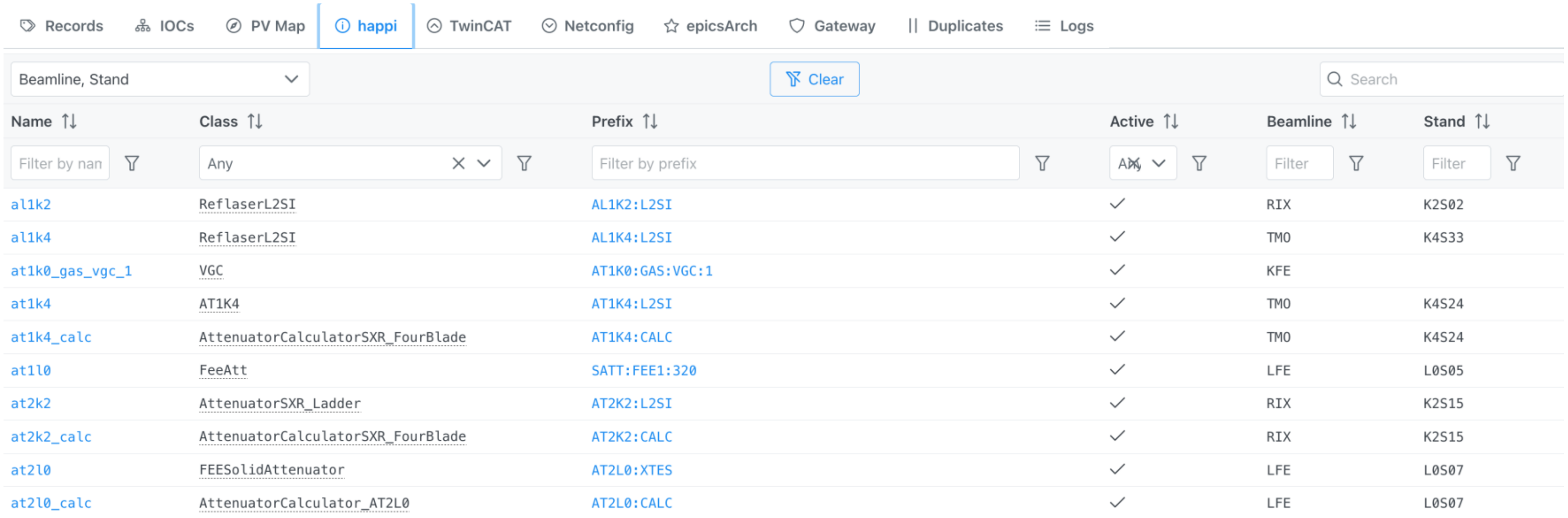
Command	Pattern	Comments	Error	File name
Filter by command	Filter by pattern	acr	Filter by error	Any
ALLOW (DEFAULT)	XCS:USER:MCC:EPHOT.*	EPHOT control for ACR		xcs.pvlist
ALLOW (DEFAULT)	LAS:UNDS:FLOAT:05	Temporary, ACR/AD/MCC Access		las.pvlist
ALLOW (DEFAULT)	EM2K0:XGMD:SHV:.*	Allow write access to XGMD HV control from soft xray control rooms and ACR		kfe.pvlist
ALLOW (DEFAULT)	MFX:USER:MCC:EPHOT.*	EPHOT control for ACR		mfx.pvlist
ALLOW (DEFAULT)	EVR:LAS:LHN:04:.*	Temporary, ACR needs access to TMO's EVR PVs		mcc.pvlist
ALLOW (DEFAULT)	TMO:USER:MCC:EPHOT.*	EPHOT control for ACR		tmo.pvlist

Duplicate records in more than one IOC



Name	IOCs
Filter by name	Filter by ioc
IOC:KFE:C:One	ioc_c ioc_d

Web frontend: happi (ophyd Device database) entries



Name	Class	Prefix	Active	Beamline	Stand
al1k2	ReflaserL2SI	AL1K2:L2SI	✓	RIX	K2S02
al1k4	ReflaserL2SI	AL1K4:L2SI	✓	TMO	K4S33
at1k0_gas_vgc_1	VGC	AT1K0:GAS:VGC:1	✓	KFE	
at1k4	AT1K4	AT1K4:L2SI	✓	TMO	K4S24
at1k4_calc	AttenuatorCalculatorSXR_FourBlade	AT1K4:CALC	✓	TMO	K4S24
at1l0	FeeAtt	SATT:FEE1:320	✓	LFE	L0S05
at2k2	AttenuatorSXR_Ladder	AT2K2:L2SI	✓	RIX	K2S15
at2k2_calc	AttenuatorCalculatorSXR_FourBlade	AT2K2:CALC	✓	RIX	K2S15
at2l0	FEESolidAttenuator	AT2L0:XTES	✓	LFE	L0S07
at2l0_calc	AttenuatorCalculator_AT2L0	AT2L0:CALC	✓	LFE	L0S07

What isn’t whatrecord?

It *isn’t* for live views: **no** PVAccess and **no** Channel Access.

As a toy/side-project with no charge code:

- It isn’t well-documented (bet you didn’t see that one coming)
  - But there are nice docstrings, generally!
- It isn’t error-free/bug-free
  - It aims to be as compliant as possible when parsing the files, but there may be discrepancies
- It isn’t a good example of how to store relational data or do web development
  - Goal was ~~breadth-first~~ whim-first:
    - Parse and interpret everything: in-memory dataclasses storing all information
    - Get it to be displayed in a friendly way
  - Database-backed information along with and corresponding backend/frontend changes may need to be pursued

Trying it out

Easiest method to try the frontend/backend as shown in the slides is with docker:

```
$ git clone https://github.com/pcdshub/whatrecord

$ cd whatrecord/docker

$ docker-compose up

# (Wait a couple minutes, then open http://localhost:8896 in browser)
```

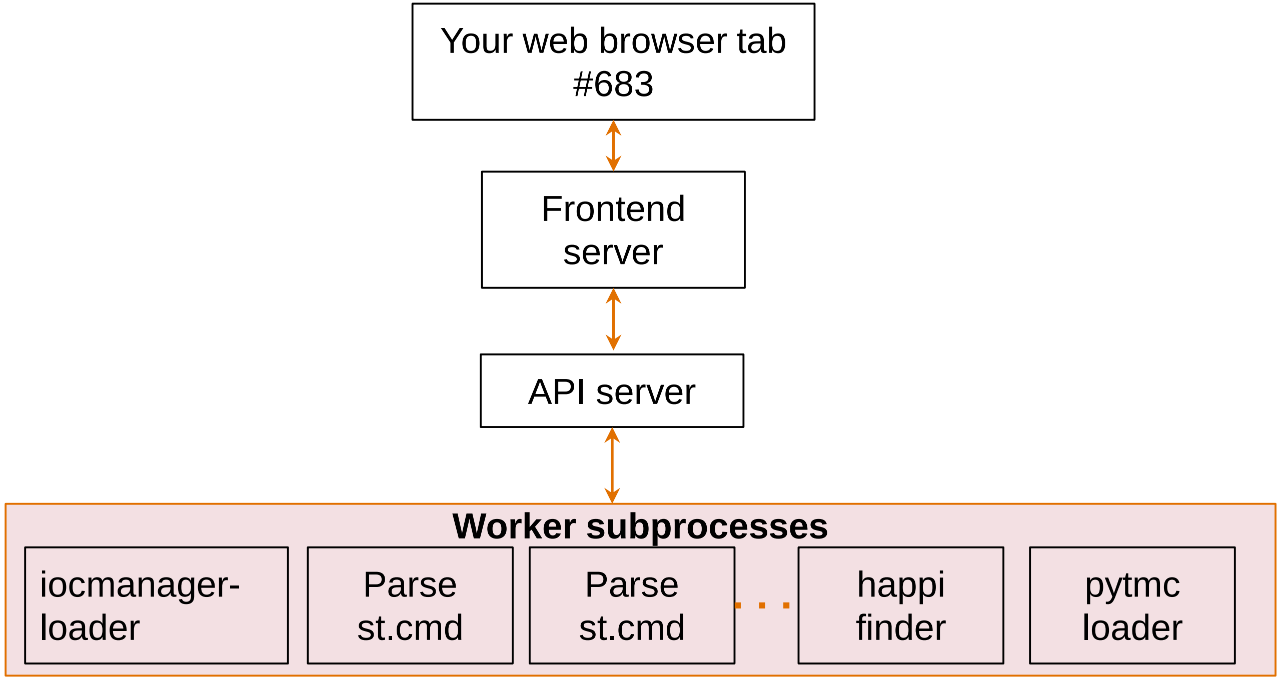
Or try the parsing tools with just Python (3.8+):

```
$ pip install whatrecord

$ whatrecord --help
```

Thank you for your time.

Backup

Your browser to the server



Grammar-based parsing?

EBNF Grammar rules - simplified excerpt from the V3 database grammar:

```
database: record*

record: "record" record_head record_body?

record_head: "(" string "," string ")"
record_body: "{" record_field* "}"

record_field: "field" "(" string "," string ")"
            | "info" "(" string "," string ")"  -> record_field_info
            | "alias" "(" string ")"            -> record_field_alias
```

Using a pure-Python parsing library “lark”:

- Take the above to make a .db file into a set of tokens.
- Take those tokens and put them in a useful data structure.

https://github.com/pcdshub/whatrecord/tree/master/whatrecord/grammar