

caputRecorder R1-4

Tim Mooney
May, 2015

This work is supported by the U.S. Department of Energy, Basic Energy Sciences, Office of Science, under contract DE-AC02-06CH11357.

Argonne National Laboratory



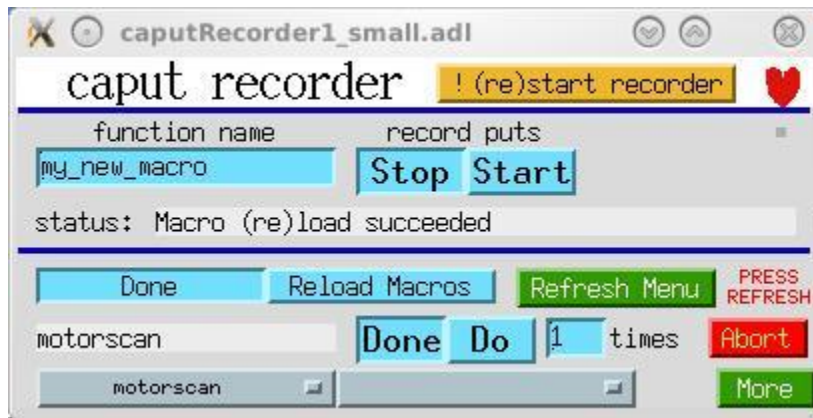
*A U.S. Department of Energy
Office of Science Laboratory
Operated by The University of Chicago*



caputRecorder

- Support for recording and playing back sequences of channel-access puts.
- User interface for executing python functions (A.K.A. “macros”)

User interface



Macro file

```

macros.py - /home/oxygen4/MOONEY/epics/synAppsSVN/supl - □ ×
File Edit Search Preferences Shell Macro Windows Help

#!/bin/env python
# examples of recorded macros (doScan) and recorded macros that have been edited
# to use arguments (initScanDo, motorscan)

import time
import epics

# The Function "_abort" is special: it's used by caputRecorder.py to abort an
# executing macro
def _abort(prefix):
    print "aborting"
    epics.caput(prefix+"AbortScans", "1")
    epics.caput(prefix+"allstop", "stop")
    epics.caput(prefix+"scaler1.CNT", "Done")

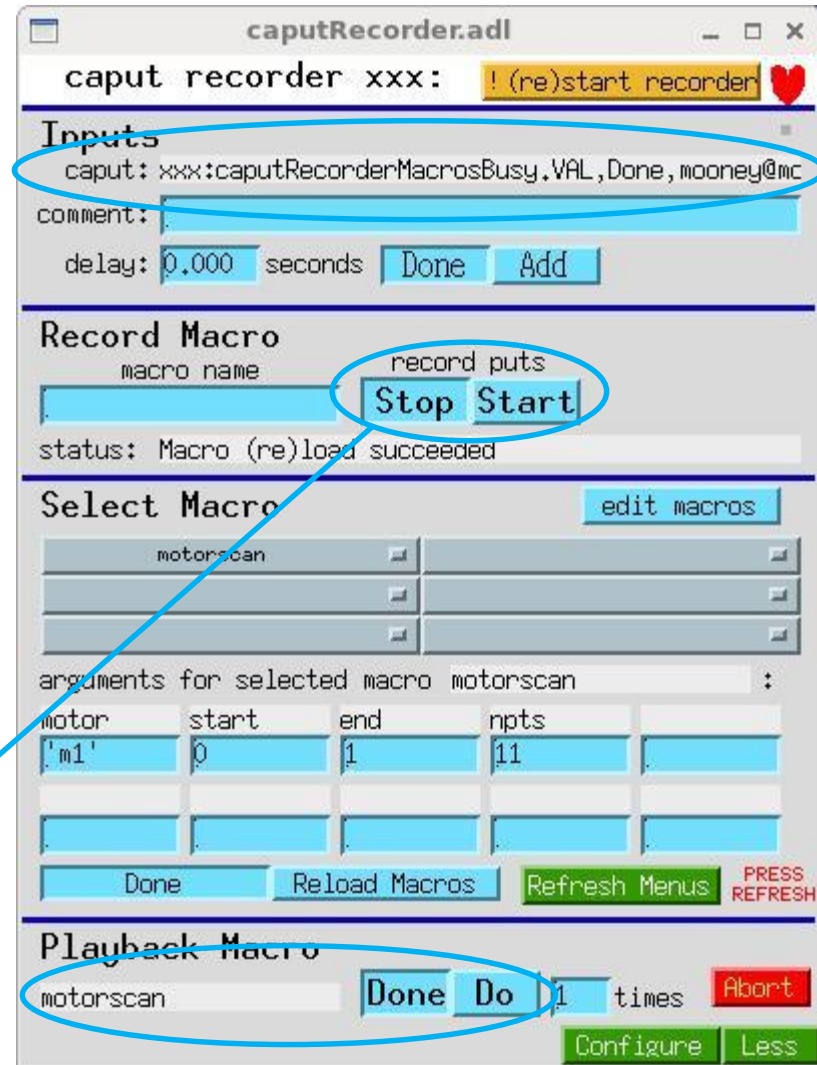
def doScan():
    recordDate = "Mon Jan  5 13:44:08 2015"
    epics.caput("xxx:scan1.P1PV","xxx:m2.VAL", wait=True, timeout=300)
    epics.caput("xxx:scan1.P1SP","0,00000", wait=True, timeout=300)
    epics.caput("xxx:scan1.P1EP","1,00000", wait=True, timeout=300)
    epics.caput("xxx:scan1.P1SI","0,10000", wait=True, timeout=300)
    epics.caput("xxx:scan1.D01PV","xxx:userCalcOut1.VAL", wait=True, timeout=300)
    epics.caput("xxx:scan1.EXSC","1", wait=True, timeout=300)

def initScanDo(start=0, end=1):
    epics.caput("xxx:scan1.NPTS","21", wait=True, timeout=300)
    epics.caput("xxx:scan1.P1PV","xxx:m1.VAL", wait=True, timeout=300)
    epics.caput("xxx:scan1.P1SP",start, wait=True, timeout=300)
    epics.caput("xxx:scan1.P1EP",end, wait=True, timeout=300)
    epics.caput("xxx:scan1.P1SM","LINEAR", wait=True, timeout=300)
    epics.caput("xxx:scan1.P1AR","ABSOLUTE", wait=True, timeout=300)
    epics.caput("xxx:scan1.P1SM","STAY", wait=True, timeout=300)
    epics.caput("xxx:scan1.T1PV","xxx:scaler1.CNT", wait=True, timeout=300)
    epics.caput("xxx:scan1.D01PV","xxx:userCalcOut1.VAL", wait=True, timeout=300)
    epics.caput("xxx:scan1.EXSC","1", wait=True, timeout=300)

def motorscan(motor="m1", start=0, end=1, step=.1):
    epics.caput("xxx:scan1.P1PV","xxx:%s.VAL" % motor), wait=True, timeout=300)
    epics.caput("xxx:scan1.P1SP",start, wait=True, timeout=300)
    epics.caput("xxx:scan1.P1EP",end, wait=True, timeout=300)
    epics.caput("xxx:scan1.P1SI",step, wait=True, timeout=300)
    # example of time delay
    time.sleep(3)
    epics.caput("xxx:scan1.EXSC","1", wait=True, timeout=300)
    
```

What caputRecorder does

- **IOC side intercepts caputs and publishes them to an EPICS PV**
 - “pvname,value,user@host”
- **Client side monitors one or more such PVs**
- **Client side interacts with an EPICS database, which**
 - hosts user-interface PVs, and
 - also serves as **client API**
- **Client side maintains *macros file***
 - Adds a macro for each recording session (Start/Stop)
 - Executes macro selected in the MEDM display



caputRecorder.adl

caput recorder xxx: ! (re)start recorder

Inputs

caput: xxx:caputRecorderMacrosBusy,VAL,Done,mooney@mc

comment:

delay: 0.000 seconds Done Add

Record Macro

macro name record puts

Stop Start

status: Macro (re)load succeeded

Select Macro edit macros

| motor | start | end | npts |
|-------|-------|-----|------|
| 'm1' | 0 | 1 | 11 |

arguments for selected macro motorscan

Done Reload Macros Refresh Menus PRESS REFRESH

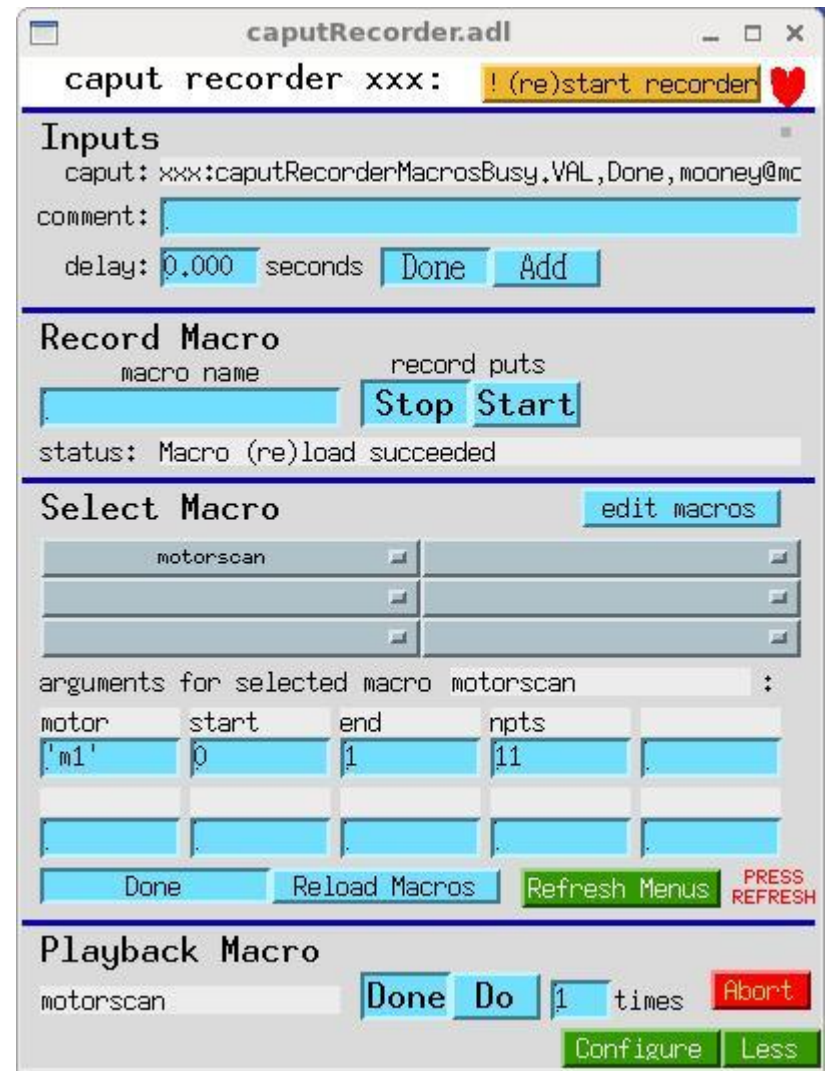
Playback Macro

motorscan Done Do 1 times Abort

Configure Less

Inputs

- **While recording, you can add:**
 - Comments
 - Time delays (`time.sleep()`)



The screenshot shows the `caputRecorder.adl` window with the following sections:

- Header:** `caput recorder xxx: ! (re)start recorder` with a heart icon.
- Inputs:**
 - caput: `xxx:caputRecorderMacrosBusy.VAL,Done,mooney@mc`
 - comment:
 - delay: seconds
- Record Macro:**
 - macro name:
 - record puts:
 - status: Macro (re)load succeeded
- Select Macro:**
 -
 - Table of macro names with checkboxes:
- arguments for selected macro motorscan:**

| motor | start | end | npts |
|----------------------|----------------------|----------------------|----------------------|
| 'm1' | 0 | 1 | 11 |
| <input type="text"/> | <input type="text"/> | <input type="text"/> | <input type="text"/> |
- Buttons:** PRESS REFRESH
- Playback Macro:**
 - motorscan times
 -

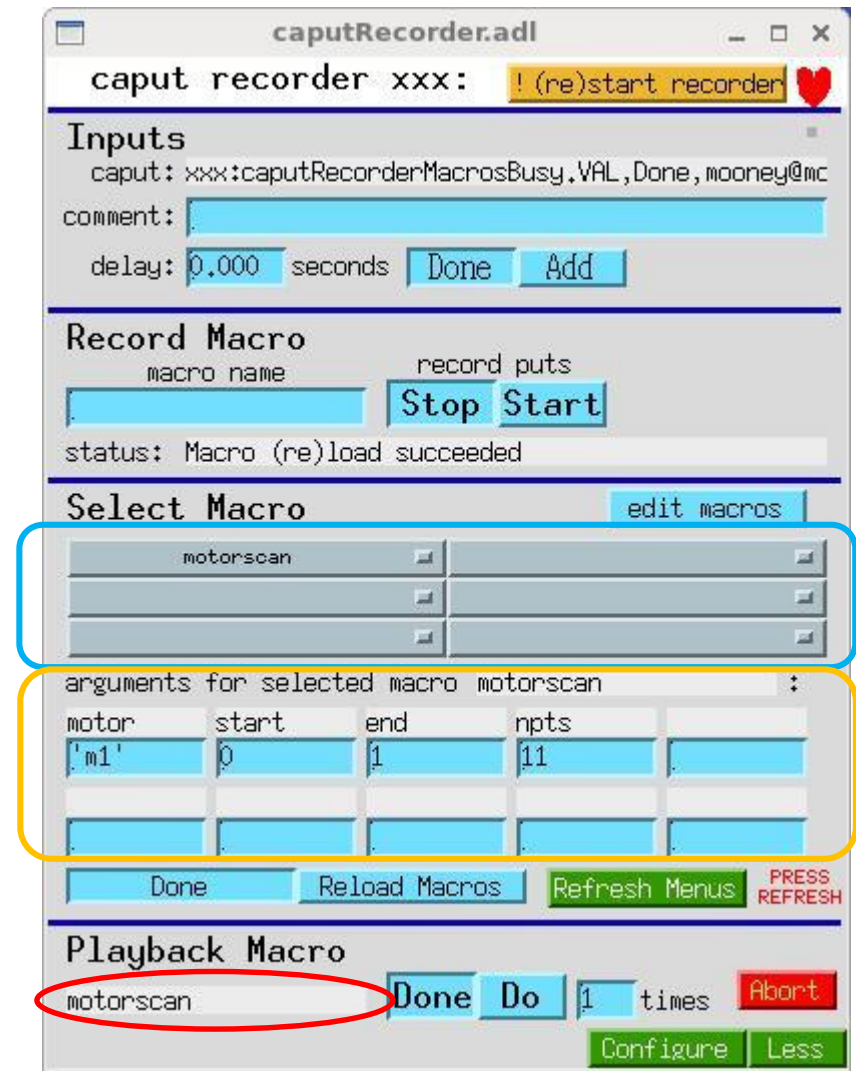
Record Macro

- **Macro name must meet python guidelines:**
 - caputRecorder will do this for you (replace illegal characters with underscores, etc.)
 - Name is truncated to 25 chars
- **When done recording, python reloads macros file, and rewrites menu PVs**
 - **BUT**, MEDM doesn't display the new menus until you press the "Refresh Menus" button



Select Macro

- Select a macro using **menu PVs**
- OR, a client can write the macro name to **xxx:caputRecorderMacro**
- caputRecorder will fill out fields for **arguments**, if any
 - As recorded, macros have no arguments, but...
 - you can edit the macros file
 - *Press “Reload Macros” afterward*
 - *If you deleted anything, press “!(re)start recorder”*
 - *Press “Refresh Menus”*



The screenshot shows the 'caputRecorder.adl' window. At the top, it displays 'caput recorder xxx:' followed by a status indicator '!(re)start recorder' and a heart icon. Below this is the 'Inputs' section with fields for 'caput: xxx:caputRecorderMacrosBusy.VAL,Done,mooney@mc' and 'comment:'. A 'delay: 0.000 seconds' field is also present with 'Done' and 'Add' buttons. The 'Record Macro' section includes a 'macro name' field and 'record puts' buttons labeled 'Stop' and 'Start'. A status message reads 'status: Macro (re)load succeeded'. The 'Select Macro' section features a table of macro names with checkboxes, and an 'edit macros' button. Below this is a table for 'arguments for selected macro motorscan' with columns for 'motor', 'start', 'end', and 'npts'. The 'Playback Macro' section shows the selected macro 'motorscan' with 'Done', 'Do', and 'Abort' buttons, and a field for the number of times to play (set to 1). A 'Configure' button and a 'Less' button are also visible.

Playback Macro

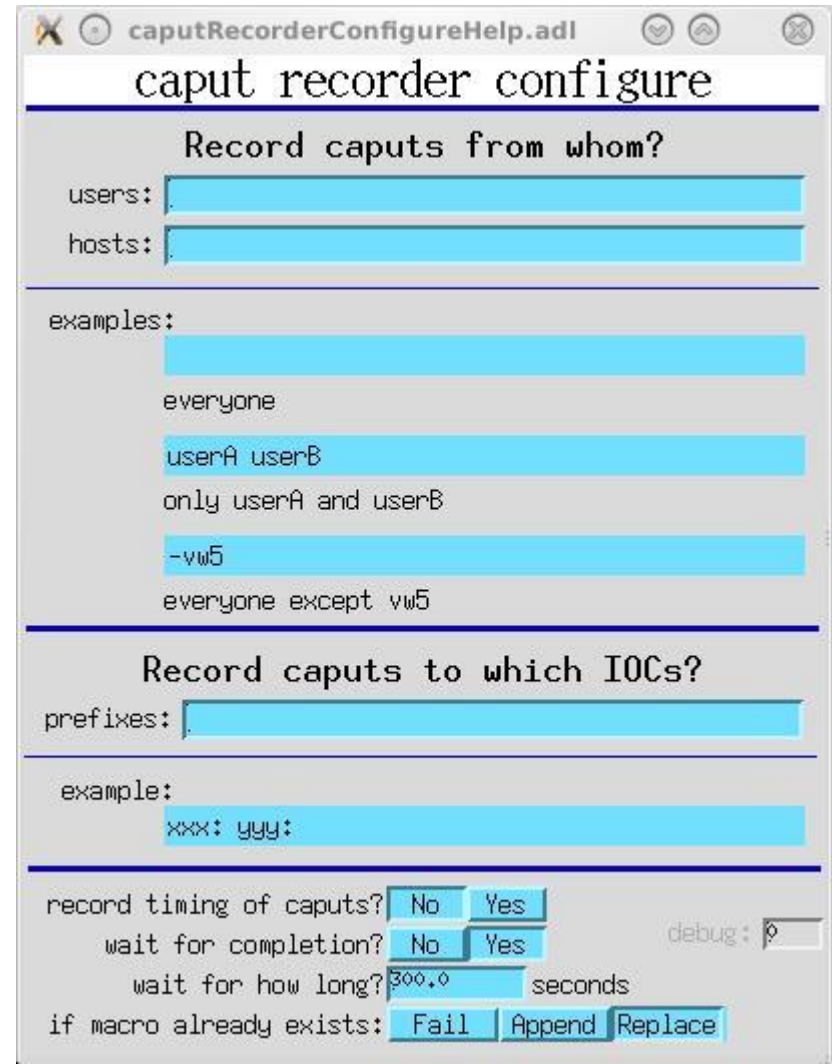
- Press “Do”
- OR, client can write to `xxx:caputRecorderExecuteMacro` with a `put_callback`
- Abort halts execution, and runs the “`_abort()`” function in the macros file
 - You should edit the `_abort()` function so it stop motors, scans, etc., for your beamline
- Note: you can add other hidden functions to the macros file:


```
def _private()
```



Configure

- **Select users and workstations**
 - Suggestion:
`users="-vw5"`
`hosts=""`
- **Select IOCs**
 - Specify the prefixes of the IOCs you want to monitor
 - Can't change this while recording
- **Can record timing of puts.**
- **Can wait for completion on replay**
 - Can specify for how long to wait.
- **If macro exists, can specify Fail/Append/Replace**
 - If **Replace**, writes backup file



The screenshot shows a window titled 'caputRecorderConfigureHelp.adl' with a subtitle 'caput recorder configure'. The main heading is 'Record caputs from whom?'. Below this are two text input fields: 'users:' and 'hosts:'. A section titled 'examples:' lists several options: 'everyone', 'userA userB', 'only userA and userB', '-vw5', and 'everyone except vw5'. The next section is 'Record caputs to which IOCs?' with a 'prefixes:' input field. An 'example:' shows 'xxx: yyy:'. At the bottom, there are several controls: 'record timing of caputs?' with 'No' and 'Yes' buttons; 'wait for completion?' with 'No' and 'Yes' buttons; 'wait for how long?' with a numeric input field set to '300.0' and the unit 'seconds'; and 'if macro already exists:' with 'Fail', 'Append', and 'Replace' buttons. A 'debug:' checkbox is also present.

Configure

