

EPICS Training

Scans

Tim Mooney
2/27/2015

Argonne National Laboratory



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



The synApps SSCAN module

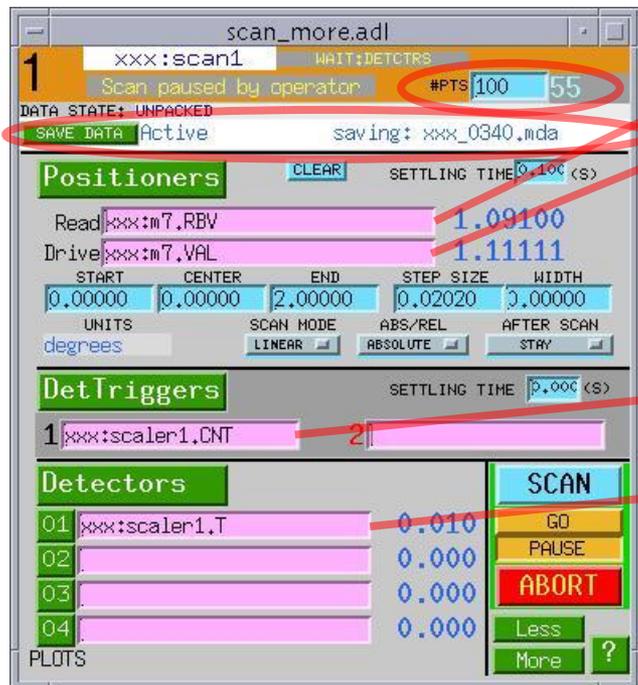
- **Where is it?**
 - <http://www.aps.anl.gov/bcda/synApps/sscan/sscan.html>
- **What's in it?**
 - Code
 - the sscan record
 - the recDynLink library
 - the saveData data-storage client
 - the scanparm record
 - EPICS databases
 - scan databases
 - scanParms and alignParms databases
 - MEDM displays
 - scan*.adl
 - scan*_help.adl

Simple scans

- A one-dimensional scan:

- Do NPTS times:
 - Set conditions
 - Trigger detectors
 - Acquire data
- Write data to disk

e.g., move motors; wait for completion
e.g., start scaler; wait for completion
read detector signals; store in arrays



scan_more.adl

1 xxx:scan1 WAIT:DETECTRS
 Scan paused by operator #PTS 100 55
 DATA STATE: UNPACKED
 SAVE DATA Active saving: xxx_0340.mda

Positioners CLEAR SETTLING TIME 0.100 (s)

Read: xxx:m7.RBV 1.09100
 Drive: xxx:m7.VAL 1.11111

START	CENTER	END	STEP SIZE	WIDTH
0.00000	0.00000	2.00000	0.02020	0.00000

UNITS: degrees SCAN MODE: LINEAR ABS/REL: ABSOLUTE AFTER SCAN: STAY

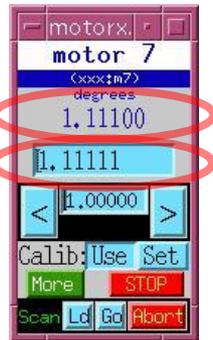
DetTriggers SETTLING TIME 0.000 (s)

1 xxx:scaler1.CNT 2

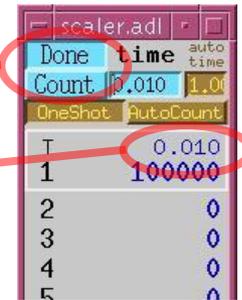
Detectors

ID	Signal	Value
01	xxx:scaler1.T	0.010
02		0.000
03		0.000
04		0.000

SCAN GO PAUSE ABORT Less More ?



motorx motor 7
 (xxx:m7) degrees
 1.11100
 1.11111
 1.00000
 Calib: Use Set
 More STOP
 Scan Ld Go Abort



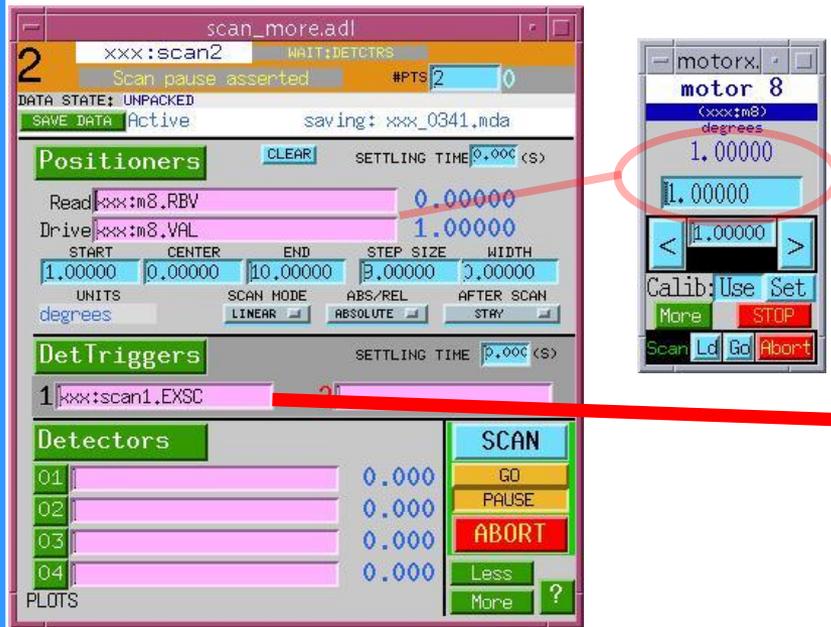
scaler.adl

Done time auto time
 Count 0.010 1.00
 OneShot AutoCount
 T 0.010
 1 100000
 2 0
 3 0
 4 0
 5 0

...Simple scans

- **Multidimensional scan:**
 - Outer-loop scan's *detector trigger* executes inner-loop scan.
 - **saveData** monitors a set of **sscan** records, determines scan dimension when scan starts, and writes data as it is acquired.
 - No limit to the number of scan dimensions.

outer-loop scan



scan_more.adl

2 xxx:scan2 WAIT:DETCRS

Scan pause asserted #PTS 2 0

DATA STATE: UNPACKED

SAVE DATA Active saving: xxx_0341.mda

Positioners CLEAR SETTTLING TIME 0.000 (s)

Read xxx:m8.RBV 0.00000

Drive xxx:m8.VAL 1.00000

START	CENTER	END	STEP SIZE	WIDTH
1.00000	0.00000	10.00000	0.00000	0.00000

UNITS degrees SCAN MODE LINEAR ABS/REL ABSOLUTE AFTER SCAN STRAY

DetTriggers SETTTLING TIME 0.000 (s)

1 xxx:scan1.EXSC 2

Detectors

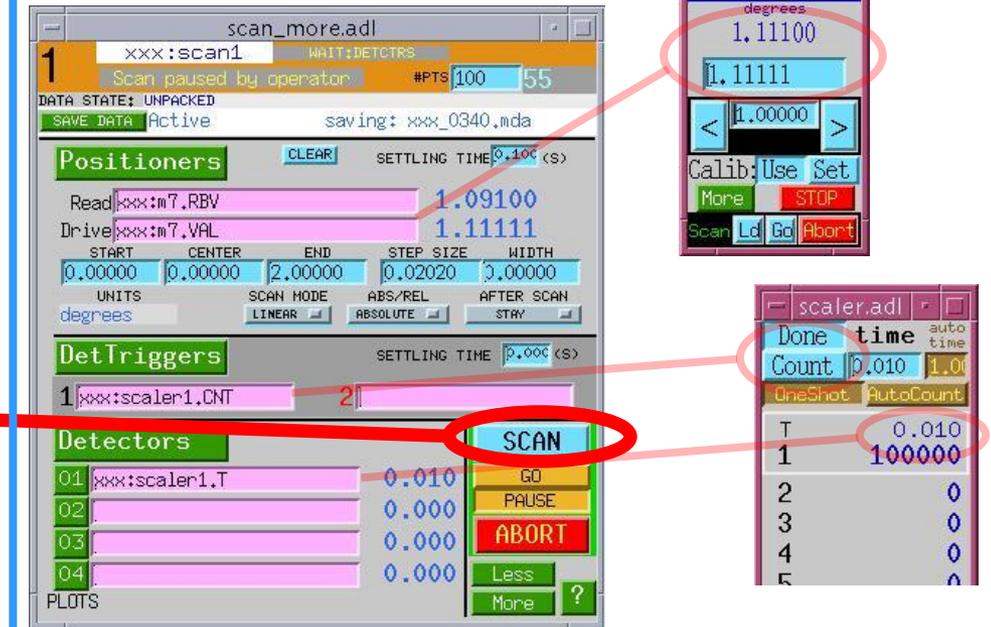
ID	Value
01	0.000
02	0.000
03	0.000
04	0.000

PLOTS

SCAN GO PAUSE ABORT Less More ?

motorx motor 8 (xxx:m8) degrees 1.00000

inner-loop scan



scan_more.adl

1 xxx:scan1 WAIT:DETCRS

Scan paused by operator #PTS 100 55

DATA STATE: UNPACKED

SAVE DATA Active saving: xxx_0340.mda

Positioners CLEAR SETTTLING TIME 0.100 (s)

Read xxx:m7.RBV 1.09100

Drive xxx:m7.VAL 1.11111

START	CENTER	END	STEP SIZE	WIDTH
0.00000	0.00000	2.00000	0.02020	0.00000

UNITS degrees SCAN MODE LINEAR ABS/REL ABSOLUTE AFTER SCAN STRAY

DetTriggers SETTTLING TIME 0.000 (s)

1 xxx:scaler1.CNT 2

Detectors

ID	Value
01	0.010
02	0.000
03	0.000
04	0.000

PLOTS

SCAN GO PAUSE ABORT Less More ?

motorx motor 7 (xxx:m7) degrees 1.11100

1.11111

1.00000

Calib: Use Set More STOP Scan Ld Go Abort

scaler.adl

Done	time	auto
Count	0.010	1.00
OneShot		AutoCount
T	0.010	
1	100000	
2	0	
3	0	
4	0	
5	0	

Scan features

- **0-4 positioners, 0-4 detector triggers, 0-70 detector signals**
 - Positioner and readback values are of type `double`
 - Detector values are of type `float`
- **Acquisition from scalar and/or array PV's**
 - Array PV's acquire `.NPTS` elements
- **Number of data points limited only by IOC memory**
 - Standard max. is 2000 (x_i, y_i) points per scan dimension
 - Can increase to $\sim \text{EPICS_CA_MAX_ARRAY_BYTES} / 8$
- **Detector/client wait, data-storage wait**
 - Can wait for multiple data-acquisition clients
 - Only one data-storage client
- **Pause/resume, abort**
 - Data from aborted scans are written to disk
- **Double buffered: writes 1D acquired data after the scan is finished**
 - Can write during next 1D scan

...Scan features

- ***saveData* writes XDR-format (".mda") files to disk.**
 - Files can be read on any type of computer
- **A positioner can have private scan parameters (scanparm record).**
 - Load preset scan parameters with one mouse click
 - Useful for alignment
- **After-scan actions include move to peak, valley, +/-edge.**
 - Can, e.g., track a moving peak through a series of scans
- **scanparm record + after-scan action = automated 1-D alignment.**

Scan implementation

- **The sscan record is a channel-access client**
 - scanned PV's can be hosted by any ioc
 - uses recDynLink library to manage connections with PV's
 - uses ca_put_callback() to set conditions, trigger detectors, and await completion
 - uses ca_get_callback() before acquiring data
- **saveData is a channel-access client**
 - monitors sscan records and user-specified PVs
 - saveData can make the sscan record wait until data are written
- **Scan acquisition/storage can run on vxWorks, Linux, or Windows.**
- **The sscan record can be driven by any channel-access client.**
 - manual operation, via MEDM, is one option
 - often driven by spec and python code
 - can simplify user-written scan-control software

Before-scan / after-scan links

- Can write a constant value to any numeric or menu PV before the scan starts and/or after the scan ends.
- Can wait or not wait for completion of processing started by the write.
- If this sscan record is part of a multidimensional scan, links function on each iteration.
- Outer-loop sscan record can write to these links, and to the values they write.
- These links can write to their own sscan record's START, END, etc. fields, but not to its link fields.



The screenshot shows the 'scan_full.adl' interface with the following sections:

- Header:** Title 'scan_full.adl', status 'IDLE', 'SCAN DIM: 0', 'OK' button, '#PTS' 1000, '0'.
- DATA STATE:** UNPACKED, 'SAVE DATA' Active, 'saveData OK'.
- BeforeScan:** Link field with value '1.0', 'BAD LINK' indicator, 'Wait' checkbox.
- Positioners:** 'ACTIVE POSITIONERS (1, , ,)', 'CHECK LIMITS', 'CLEAR POSITIONERS' buttons. Fields for 'Read' (0.000000) and 'Drive' (xxx:userCalc1,A, 0.000000). A table with columns: START, CENTER, END, STEP SIZE, WIDTH. Values: 0.0010000, 0.0000000, 1.0000000, 0.0010000, 0.0000000. Below are 'UNITS', 'SCAN MODE' (LINEAR), 'ABS/REL' (ABSOLUTE), 'AFTER SCAN' (STAY).
- DetTriggers:** 'SETTLING TIME 0.100 (S)', 'REFERENCE DETECTOR 1'. Table with columns: VAL, SETTling TIME, VAL. Values: 1.0, 0.000, 1.0, 1.0, 0.000, 1.0.
- CLIENT WAIT:** - + 0, 'AUTO WAIT FOR 0 CLIENTS'.
- Detectors:** 'DIMENSION SCALAR', 'ACQ MODE NORMAL', 'SCAN' button. Table with columns: DIMENSION, ACQ MODE, VAL. Values: 01 xxx:userCalc1.VAL, 0.000, 02, 0.000, 03, 0.000, 04, 0.000.
- PLOTS:** 'ArrayTrig' link field with value '1.0', 'BAD LINK' indicator, 'AfterScan' link field with value '1.0', 'BAD LINK' indicator, 'Wait' checkbox.
- Buttons:** 'GO', 'PAUSE', 'RESUME' (1.000 DELAY), 'ABORT', 'Less', '?'.

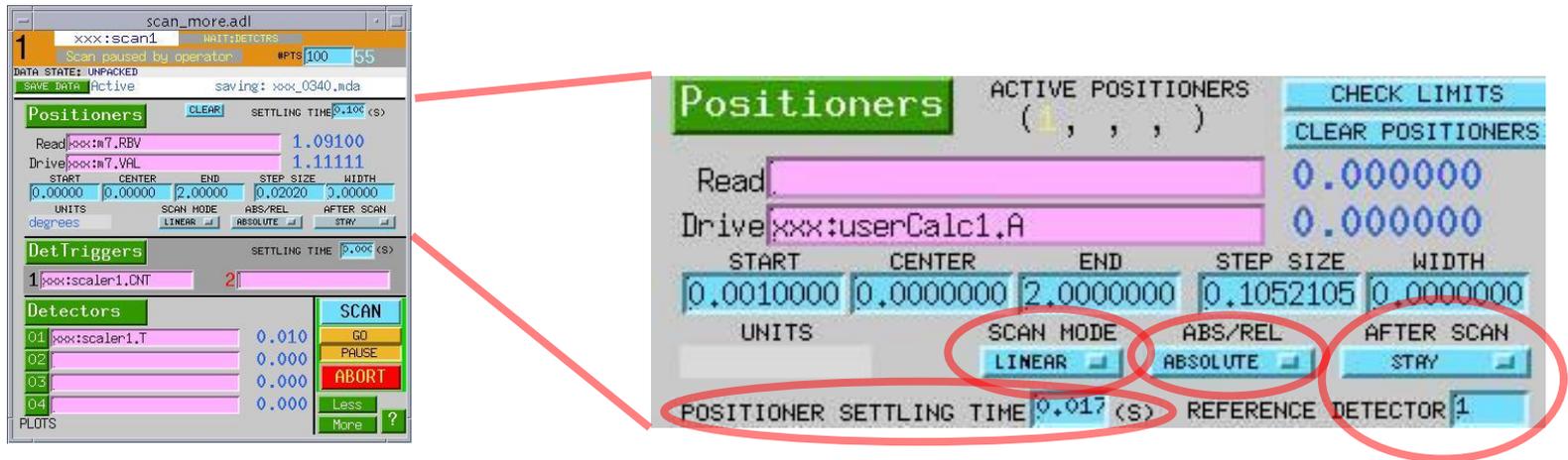


MEDM user interface

The image displays four overlapping MEDM user interface windows for scan control:

- scan.adl (top left):** Shows 'xxx:scan2' in 'IDLE' state. Parameters include #PTS: 2, Scan pause rescinded, and Scan saved: xxx_0046.mda_01. It features a 'Positioners' section with Read (xxx:m8.RBV) at 0.00000 and Drive (xxx:m8.VAL) at 1.00000. A table shows scan parameters: START (1.00000), END (10.00000), STEP SIZE (9.00000), ABS/REL (ABSOLUTE), and AFTER SCAN (STAY). A 'Detectors' section shows four detectors (01-04) at 0.000. Control buttons include SCAN, GO, PAUSE, ABORT, and More ?.
- readl (top middle):** Shows 'xxx:scan1' in 'IDLE' state. Parameters include #PTS: 2, Scan saved: xxx_0046.mda_01. It features a 'Positioners' section with Read (xxx:m8.RBV) at 0.00000 and Drive (xxx:m8.VAL) at 1.00000. A table shows scan parameters: START (1.00000), END (10.00000), STEP SIZE (9.00000), ABS/REL (ABSOLUTE), and AFTER SCAN (STAY). A 'Detectors' section shows four detectors (01-04) at 0.000. Control buttons include SCAN, GO, PAUSE, ABORT, and More ?.
- scan_full.adl (top right):** Shows 'xxx:scan1' in 'IDLE' state. Parameters include #PTS: 1000, OK, and saveData OK. It features a 'Positioners' section with Read (xxx:userCalc1.A) at 0.00000 and Drive (xxx:userCalc1.A) at 0.00000. A table shows scan parameters: START (0.0010000), CENTER (0.0000000), END (1.0000000), STEP SIZE (0.0010000), and WIDTH (0.0000000). A 'Detectors' section shows four detectors (01-04) at 0.000. Control buttons include SCAN, GO, PAUSE, ABORT, and More ?.
- scan_full.adl (bottom right):** Shows 'xxx:scan1' in 'IDLE' state. Parameters include #PTS: 1000, OK, and saveData OK. It features a 'Positioners' section with Read (xxx:userCalc1.A) at 0.00000 and Drive (xxx:userCalc1.A) at 0.00000. A table shows scan parameters: START (0.0010000), CENTER (0.0000000), END (1.0000000), STEP SIZE (0.0010000), and WIDTH (0.0000000). A 'Detectors' section shows four detectors (01-04) at 0.000. Control buttons include SCAN, GO, PAUSE, ABORT, and More ?.

Positioner options



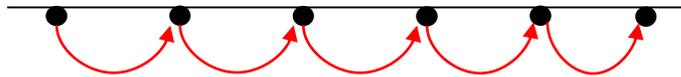
- **SCAN MODE (.PnSM - per positioner)**
 - Determines how and to where positioner moves
- **Absolute/Relative (.PnAR - per positioner)**
 - Determines how positioner locations are written
- **Positioner delay (.PDLY - affects all positioners)**
 - Delay while positioners are settling, after completing their moves
- **After-scan motion (.PASM - affects all positioners)**
 - Determines what, if anything, is done with positioners when scan is finished

...Positioner options

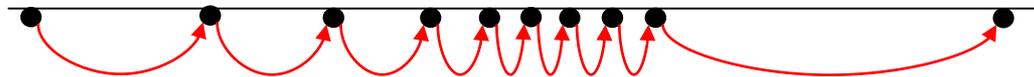
- **SCAN MODE (.PnSM - per positioner)**
 - **LINEAR** – Evenly spaced positions are calculated algorithmically
 - You specify positioner locations by setting **any three** of

<i>START</i>	<i>CENTER</i>	<i>END</i>	<i>WIDTH</i>	<i>STEP SIZE</i>	<i># POINTS</i>
.PnSP	.PnCP	.PnEP	.PnWD	.PnSI	.NPTS

- The sscan record reconciles unset parameters

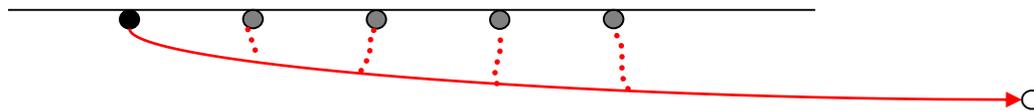


- **TABLE** – Positioner locations are contained in the **.PnPA** array
 - The array must contain at least **.NPTS** values
 - You must arrange for the array to contain the desired positions before starting the scan.
 - The **.PnPA** array is never overwritten by the sscan record



...Positioner options

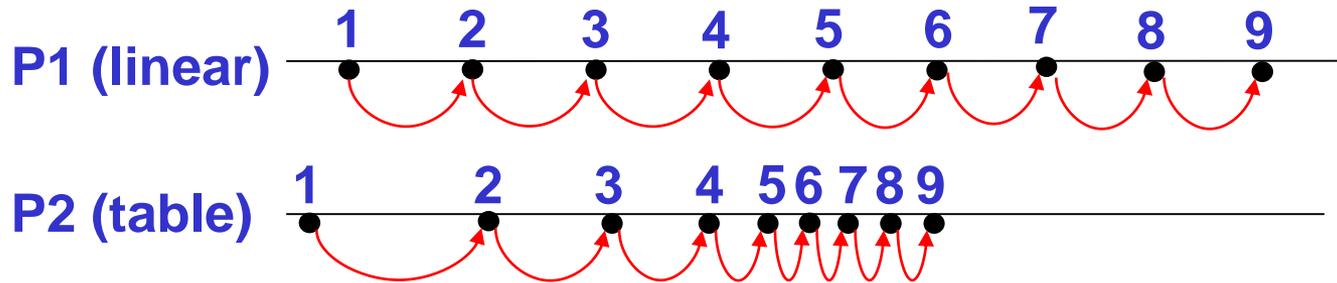
- ...SCAN MODE (.PnSM - per positioner)
 - FLY – data will be acquired *while* positioner moves
 - You specify positions at which data are acquired by setting *START*, *END*, positioner speed, and detector acquisition time.
 - The following algorithm is executed:
 - Positioner sent to *START*; reports completion
 - Detector triggered; reports completion
 - First data point acquired
 - Positioner sent to *END*
 - *NPTS-1* iterations of
 - Detector triggered; reports completion
 - Data point acquired
 - The timing of data points is controlled by the detector's acquisition time.
 - Fly-mode positioners do not report completion. (The positioner may still be moving after the scan ends.)
 - Note: timing of readback from a fly-mode positioner is inexact



...Positioner options

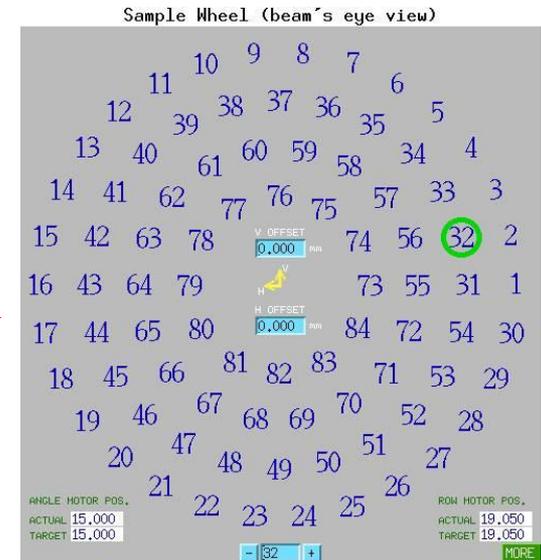
- ...SCAN MODE (.PnSM - per positioner)

- OK to mix scan modes:



- Don't be limited by existing positioner modes

- A positioner is *anything* you can write to
- Can specify positions algorithmically, using calcout or transform
 - E.g., sample-wheel →
- Can write to positioner through interpolation table
 - Use a spare positioner readback to get actual positions into the data file



...Positioner options

- **Absolute/Relative (.PnAR - per positioner)**
 - If **.PnAR** == “ABSOLUTE” (0), positions are sent exactly as given.
 - If **.PnAR** == “RELATIVE” (1), positions are added to pre-scan position before being sent to positioner.

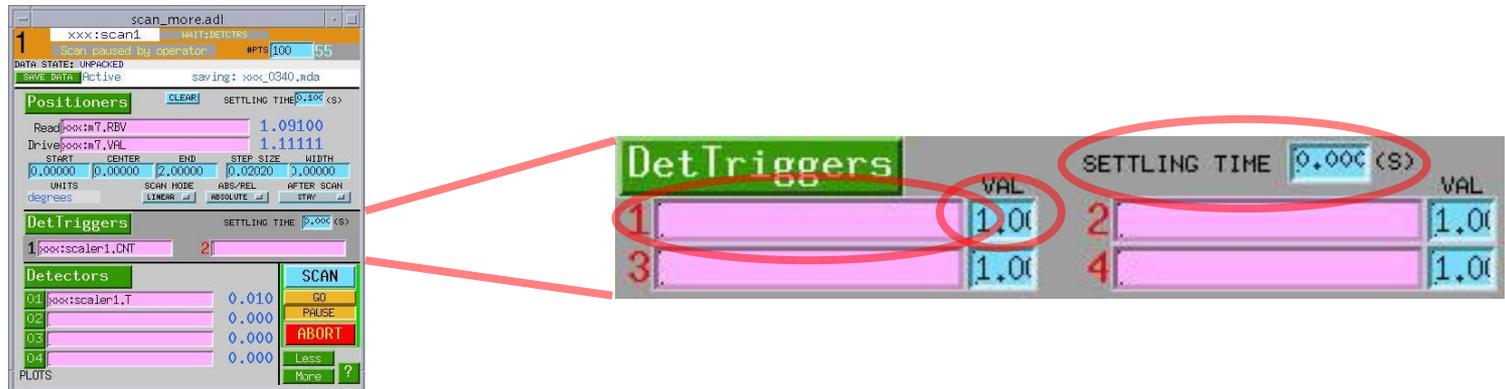
- **Settling time (.PDLY - affects all positioners)**
 - If any positioner PV is specified, then after all positioners report completion, the sscan record waits for **.PDLY** seconds before moving to next phase of sscan.
 - Useful for positioners that “ring” after move is completed
 - Useful work-around for positioners that cannot report completion
 - If no positioners, then settling time is ignored.
 - Settling time is adjusted to nearest multiple of system-clock period (typically 1/60Hz).

...Positioner options

- **After-scan motion (.PASM - affects all positioners)**
 - STAY – positioners are simply left where they ended up
 - START POS – positioners are sent to their *START* positions
 - PRIOR POS – positioners are sent to their pre-scan positions

 - PEAK POS – data from the reference detector (number given by the **.REFD** field, in range [1..70]) is examined. If a peak is found, positioners are sent to where it was acquired.
 - VALLEY POS – similar, but valley instead of peak
 - +EDGE POS – peak of derivative of reference data
 - -EDGE POS – valley of derivative of reference data
 - CNTR OF MASS – center of mass: $\Sigma x_i y_i \Delta x_i / \Sigma y_i \Delta x_i$. If more than one positioner is active, data from the lowest numbered active positioner will be used in the calculation. The result will, nevertheless, be applied to all active positioners.

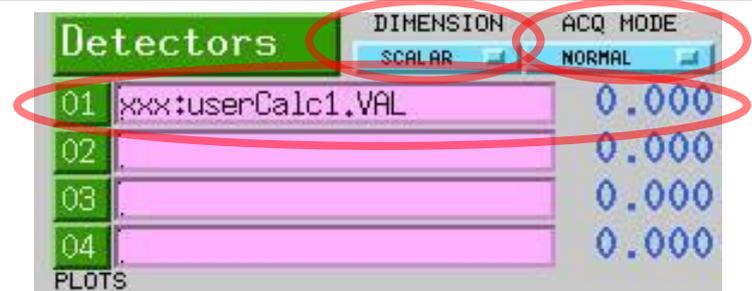
Detector triggers



- 0-4 detector triggers (.TnPV), intended to start data-acquisition
- Similar to positioners, but value sent (.TnCD) is constant
- Triggers execute after all positioners have completed, and after any positioner settling time has elapsed.
- Detector settling time begins after all detector triggers have reported completion.
- If no triggers, then settling time is ignored.

Detectors

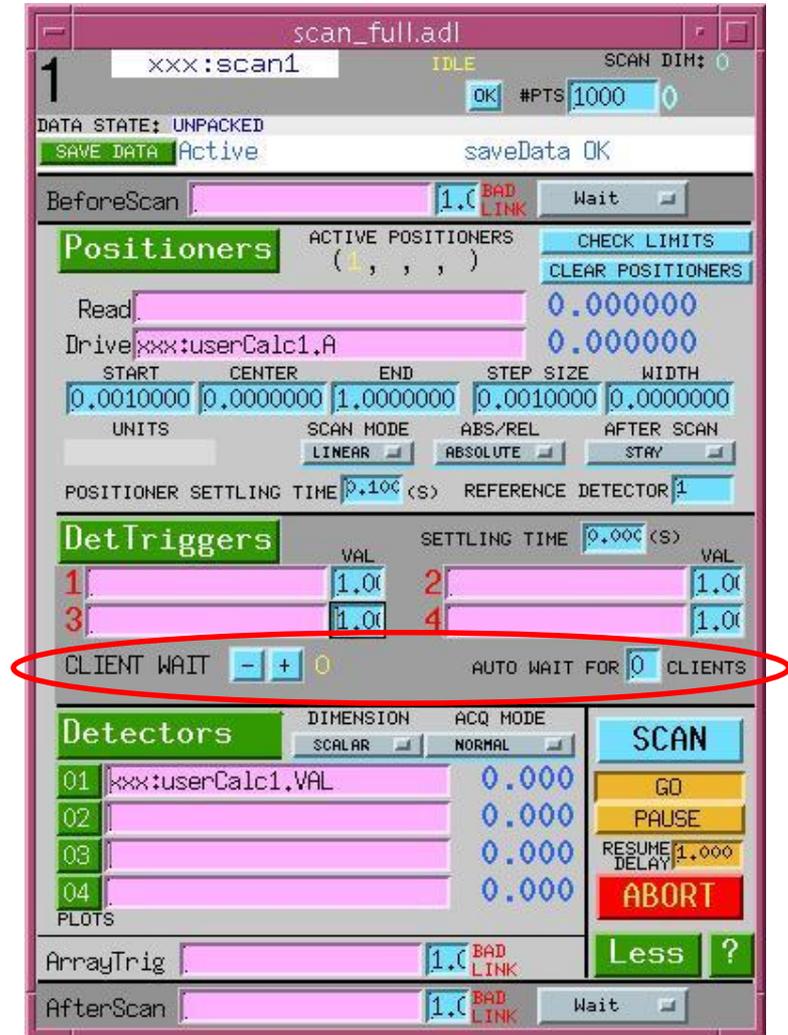
- PV's to be acquired during scan
- 0-70 detectors (.D01PV - .D70PV)
- Detector options



- Acquisition type (.ACQT)
 - SCALAR
 - scalar PV's acquired at each positioner location
 - Array PV's (.NPTS elements) acquired at end of scan
 - 1D ARRAY
 - use this mode only if ALL detectors are array valued
 - Positioners are only sent to their START positions.
 - In the future, array-valued positioners may be supported.
- Acquisition mode (.ACQM)
 - NORMAL – store values as acquired
 - ACCUMULATE – add detector values, starting with next scan
 - ADD TO PREV – same, but starting with previous scan

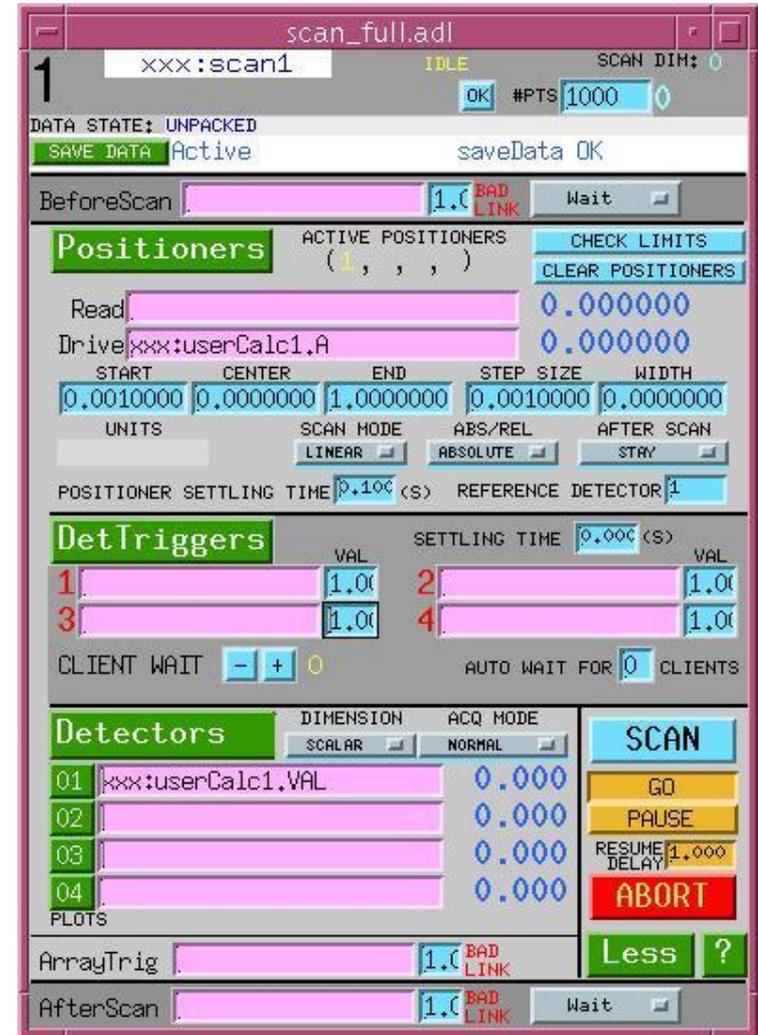
Client wait

- After all detector triggers have reported completion, and before acquiring data, the sscan record checks for client waits.
- Clients can hold scan at this point by writing '1' to .WAIT (this increments the wait-count field, .WCNT)
- Several clients can use this field
- When all clients have written '0' to .WAIT, scan acquires data.
- If clients are too slow to write to .WAIT, scan can set .WCNT for them, to the value .AWCT.
- Scan will pause until .AWCT clients have written '0' to .WAIT.
- 'Client' includes user, via MEDM



Array trigger/wait

- After all data points have been acquired, scan can trigger software that prepares array PVs for acquisition (e.g., read from hardware).
- When array trigger declares completion, array PVs are acquired.



The screenshot shows the EPICS scan configuration window for 'scan_full.adl'. The scan is named 'xxx:scan1' and is currently in an 'IDLE' state. The data state is 'UNPACKED' and the 'SAVE DATA' option is 'Active'. The scan parameters are set to 1000 points. The 'BeforeScan' section shows a 'Wait' option with a value of 1.0. The 'Positioners' section shows 'ACTIVE POSITIONERS' with a value of (1, , ,). The 'Read' and 'Drive' fields are set to 'xxx:userCalc1.A'. The 'START', 'CENTER', 'END', 'STEP SIZE', and 'WIDTH' fields are set to 0.0010000, 0.0000000, 1.0000000, 0.0010000, and 0.0000000 respectively. The 'SCAN MODE' is 'LINEAR', 'ABS/REL' is 'ABSOLUTE', and 'AFTER SCAN' is 'STAY'. The 'POSITIONER SETTLING TIME' is 0.100 (s) and the 'REFERENCE DETECTOR' is 1. The 'DetTriggers' section shows four triggers with 'VAL' set to 1.0 and 'SETTLING TIME' set to 0.000 (s). The 'CLIENT WAIT' is 0 and 'AUTO WAIT FOR' is 0 CLIENTS. The 'Detectors' section shows four detectors with 'DIMENSION' set to 'SCALAR' and 'ACQ MODE' set to 'NORMAL'. The 'SCAN' section has buttons for 'SCAN', 'GO', 'PAUSE', 'RESUME', 'DELAY', and 'ABORT'. The 'ArrayTrig' and 'AfterScan' fields are set to 'xxx:userCalc1.A' with a value of 1.0. A blue arrow points to the 'ArrayTrig' field.

Scan controls

- **SCAN**

- Writing '1' starts this sscan record
- Writing '0' stops this sscan record. (But with the supplied database, always use the 'ABORT' button to stop.)

- **GO/PAUSE**

- Pause is immediate, Go occurs after delay

- **ABORT**

- Writes '1' to 'xxx:allstop.VAL', which should stop motors
- Sends "stop" message to *all* sscan records in the supplied database
 - First 'Abort' attempt ends scan after outstanding completion callbacks have come in, and data-storage client has released the previous scan's data arrays.
 - Second 'Abort' attempt waits only for data-storage client.
 - Third successive 'Abort' attempt kills scan with no regard for consequences.



Scan user documentation

MEDM displays

scan_help.adl

This display controls a one-dimensional scan, or one dimension of a multi-dimensional scan. In multi-dimensional scans, inner scans act as detectors (triggered and waited for, though usually not read) for outer scans.

In a one-dimensional scan, the following sequence of actions occurs:

- 1) Write to before-scan PV, and (option) wait for completion.
- 2) Generally, NPTS iterations of:
 - Write to positioners.
 - Wait for positioners to declare themselves 'done'.
 - Write to detector triggers.
 - Wait for triggers to declare 'done'.
 - Wait until client-wait count is zero.
 - Wait for detector-settling time.
 - Read positioner readbacks and detectors. If acquisition mode is 'ACCUMULATE' or 'ADD-TO-PREV', new data is added to last scan's data.
- 3) Write to array-read trigger, wait for completion, and read any array-valued detector signals.
- 4) Wait for data-storage client to finish writing last scan's data.

See also: [About Display Fields](#)

Mostly status and identifying info. The number of data points is also specified here.

State of data according to scan record 'POSTED' means it's been sent to saveData.

Info about saveData -- the program that monitors scans and writes data files.

Command that executes before any positioners move. Usually, a write causes some processing to occur, and you choose whether the scan should wait for that processing to complete before proceeding to the next step.

Positioners set conditions under which data will be acquired. For examples, you can move motors, set amplifier gains, etc. After all positioner commands have been sent, the scan waits for positioners to declare themselves done, and then waits for any programmed settling time, before triggering detectors. [MORE](#)

Detector triggers are like positioners, but they are intended to start data acquisition, and the values written to them do not vary during a scan. Because some data-acquisition is done by clients that can't declare completion in the normal way, a client-wait field allows clients to declare completion by writing to a PV. If clients aren't quick enough to resign completion, the software can do it for them. [MORE](#)

Detectors are signals read after all triggers and data-acquisition clients have completed. Any readable, numeric PV can be named as a detector. If an array-valued PV is named, NPTS elements will be acquired. (If all PV's are array valued, you can set the acquisition type ('DIMENSION') to '1D ARRAY'.) [MORE](#)

Command that executes after the scan is finished, data has been posted, and any positioner-after-scan motions have completed.

scan_detector_help.adl

Detector triggers are like positioners, but they are intended to start data acquisition, and the values written to them do not vary during a scan. A writable PV can be named as a detector trigger, but if the scan is to wait for the triggered action to complete, then either the PV must be implemented so that it's 'done' callback results does signify completion of that action, or some other wait mechanism must be used.

The possibility is the detector-settling time. If no better mechanism is handy, you can just make the settling time long enough to ensure that acquisition is done before the scan reads the data.

A better mechanism is for the data-acquisition client to declare completion by writing '0' to the scan's WAIT field. (The CLIENT-WAIT button labels). See exactly this.) The WAIT mechanism is a timeout hardware, and the client normally asserts that it's busy by writing '1' to WAIT (the '1' button does this) and then asserts 'done' by writing '0'.

If the client can assert 'done', but can't assert 'busy' (or might not do it quickly enough), the scan can assert 'busy' on the client's behalf. To arrange this, simply set the AUTO-WAIT count to the number of clients that will assert 'done' but will not assert 'busy'. Just make sure the scan and the client don't BOTH assert 'busy' for the same operation.

Detectors are signals read after all triggers and data-acquisition clients have completed. Any readable, numeric PV can be named as a detector. If an array-valued PV is named, NPTS elements will be acquired. (If all PV's are array valued, you can set the acquisition type ('DIMENSION') to '1D ARRAY'.)

It's ok to have a scan with no detector triggers, no data-acquisition clients, no detectors, or any combination of the above. If the scan doesn't have to wait for trigger callbacks or data-acquisition clients, then it won't delay for the triggered action to complete, and then either the PV must be implemented so that it's 'done' callback results does signify completion of that action, or some other wait mechanism must be used.

The 'reference detector' is the detector whose data will be used to direct positioner after-scan motions. For example, positioners can be sent to the peak after a scan is finished.)

Data can be saved over several scan iterations. To start such an accumulation beginning with the next scan, set the acquisition mode to 'ACCUMULATE'. To include already acquired data, set the acquisition mode to 'ADD-TO-PREV'.

Command that executes after all scalar data acquisition is done. Intended to cause any array hardware to post data.

scan_help.adl

Command that executes after all scalar acquisition is done. Intended to cause any array hardware to post data.

NPTS - The number of data points to be acquired in (one loop of) this scan (dimension). NPTS applies to all positioners.

ABS/REL - controls whether positions generated by algorithm are to be acquired as generated or added to pre-scan positions. Each positioner has its own ABS/REL parameter.

AFTER-SCAN - controls the values written to positioners after the last data point has been acquired. AFTER-SCAN applies to all positioners. The choices:

- STOP - No values are written.
- STAFF PCD - Positioners are sent to their first-data-point positions.
- PREP PCD - Positioners are sent to their pre-scan positions.
- POST PCD - Data acquired from the reference detector is assumed, and positioners are sent to positions at which the peak value occurred.
- VELLY PCD - The positions at which the smallest data value was acquired.
- MODE - Peak of the derivative of reference-detector data.
- MODE - Valley of the derivative.

In FLY mode, the timing of data acquisition is controlled entirely by detector dwell times, and it is your job to make sure that positioners move at speeds commensurate with dwell times. Scan software will acquire positioner-readback data just before it acquires detector-signal data, but otherwise it regards flymode positioners as leeches on cables.

It's ok for different positioners to have different scan modes.

If any positioners are in LINEAR or TRAIL mode, the scan waits after all positioners have declared themselves done, for the positioner-settling time to elapse, before triggering detectors. If no positioners are in LINEAR or TRAIL mode, the positioner-settling time is moot.

It's ok to have a scan in which no positioners are specified at all. In this case, none of the positioner parameters matter. If positioner-readbacks are specified, they will still be read.

Just to make sure you're sure of any positioners that have been specified, but that don't show up on this display.

A PV has been specified, but no connection exists to the PV. This might mean the IOC hosting the PV is not running, or the PV is misspelled, or there is network trouble.

'NPTS' - See if the scan would violate any positioner limits.

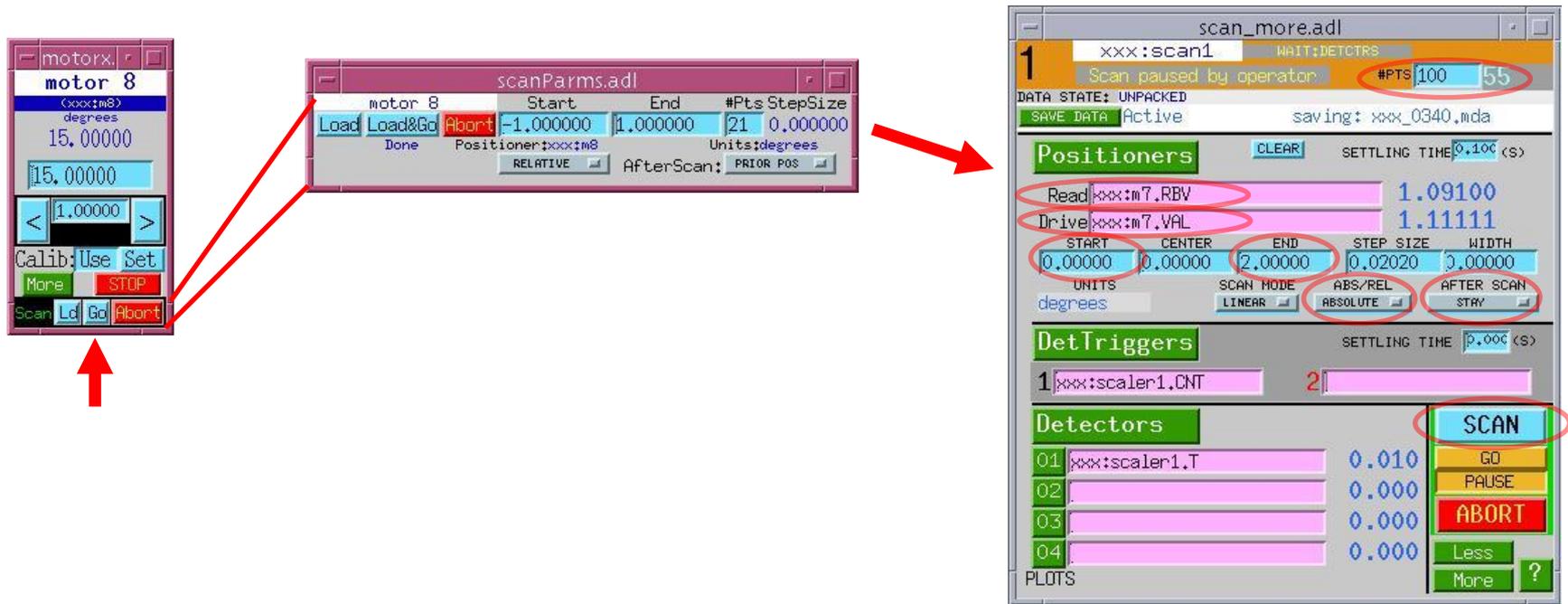
Erase all positioner PV names.

See description of the AFTER-SCAN mode

Rounded to the nearest clock 'tick'. Typically, a 90-Hz clock is used.

One-click scans

- The scanparm record executes preprogrammed *linear* scans
 - Holds scan parameters for a positioner
 - Writes parameters to a particular **sscan** record
 - Optionally executes the **sscan** record
 - Useful for alignment

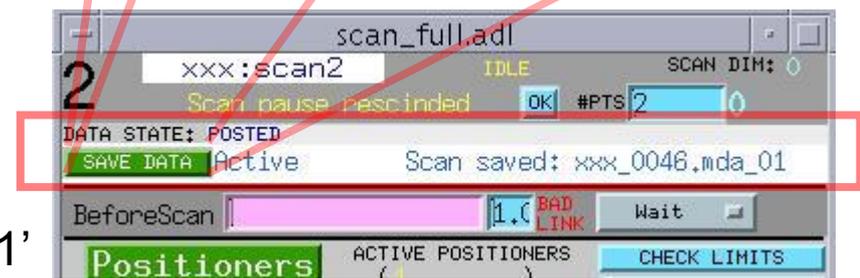
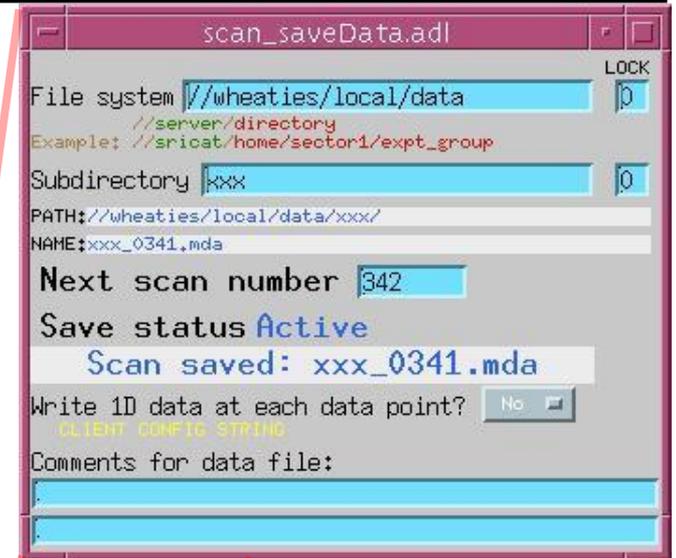


The image shows three windows from the EPICS control interface:

- motor 8**: Shows a position of 15.00000 degrees. A red arrow points to the 'Scan' button.
- scanParms.adl**: Shows parameters for 'motor 8'. The 'Start' is -1.000000, 'End' is 1.000000, and '#Pts' is 21. A red arrow points from this window to the scan_more.adl window.
- scan_more.adl**: Shows the execution of 'xxx:scan1'. The status is 'Scan paused by operator'. The '#PTS' is 100 and '55'. The 'Positioners' section shows 'Read:xxx:m7.RBV' (1.09100) and 'Drive:xxx:m7.VAL' (1.11111). The 'START' (0.00000), 'CENTER' (0.00000), and 'END' (2.00000) values are circled in red. The 'SCAN MODE' is 'LINEAR', 'ABS/REL' is 'ABSOLUTE', and 'AFTER SCAN' is 'STAY'. The 'DetTriggers' section shows '1|xxx:scaler1.CNT' and '2|'. The 'Detectors' section shows '01|xxx:scaler1.T' with a width of 0.010. A red circle highlights the 'SCAN' button.

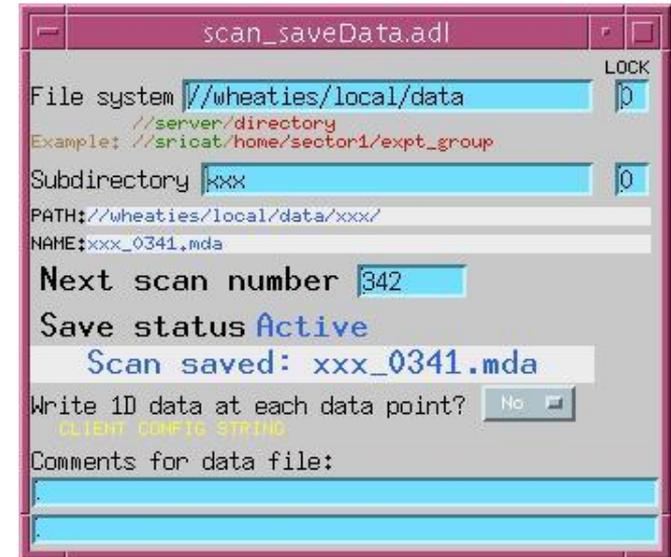
Data storage

- saveData monitors sscan records and writes their data to numbered files.
- Handshake permits pipelined operation.
- saveData's boot-time init can specify list of PV's to write with every scan's data
- saveData writes "MDA" files
 - MultiDimensional Archive
 - Binary, cross-platform (XDR) format
 - Format is optimized for run-time access.
 - Format permits file to be closed after each set of writes.
- Automatic file numbering
 - e.g., 'xxx_0123.mda', 'xxx_0124.mda'
 - overlap is handled: 'xxx_0123.mda_01'



...Data storage

- **Location of data files**
 - ‘File system’ + ‘subdirectory’
 - vxWorks:
 - File system is NFS-mount point
 - ‘//<hostname>’ is required
 - Linux, etc.:
 - saveData doesn’t mount the file system (system administrator does this)
 - ‘//<hostname>’, if present, is ignored
- **Cannot write to ‘File system’ or ‘subdirectory’ while a scan is in progress. (See ‘LOCK’ PV.)**
- **Don’t delete or rename the directory saveData is writing to.**
- **Comment PV’s saved only if they are named in saveData.req**



saveData.req init file

```

[prefix] ←
$(P)

[status]
$(P) saveData_status
...

[scanRecord] ←
$(P) scanH
$(P) scan1
$(P) scan2
$(P) scan3
$(P) scan4

[extraPV] ←
#<PV name> <description>
$(P) scaler1.TP "scaler preset (s)" ←
$(P) scaler1.NM1 "scaler chan 1 desc"
...

```

Section head

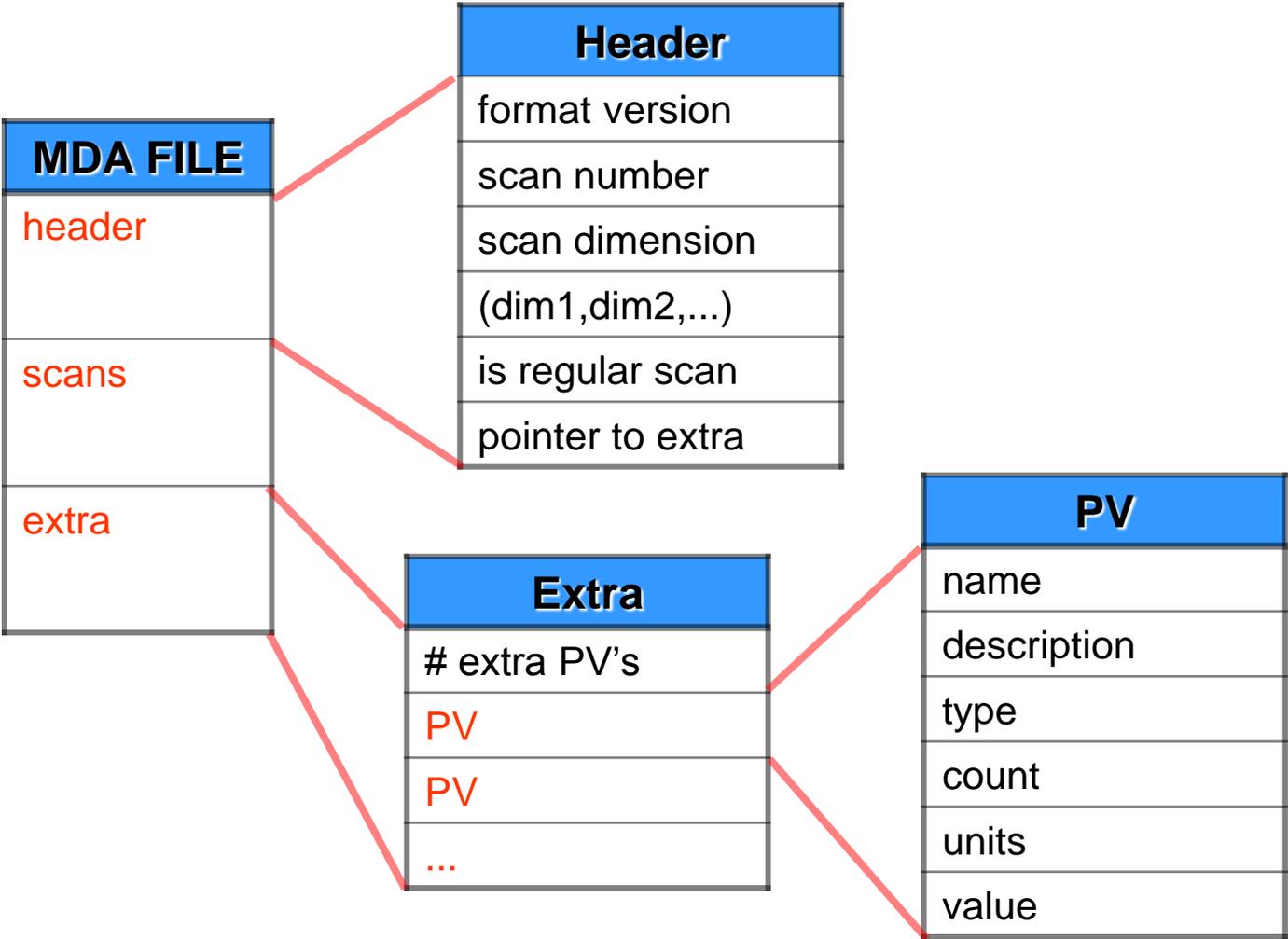
List of *sscan* records to monitor

List of PV's to be saved with every scan (Normally, this is the only section you modify.)

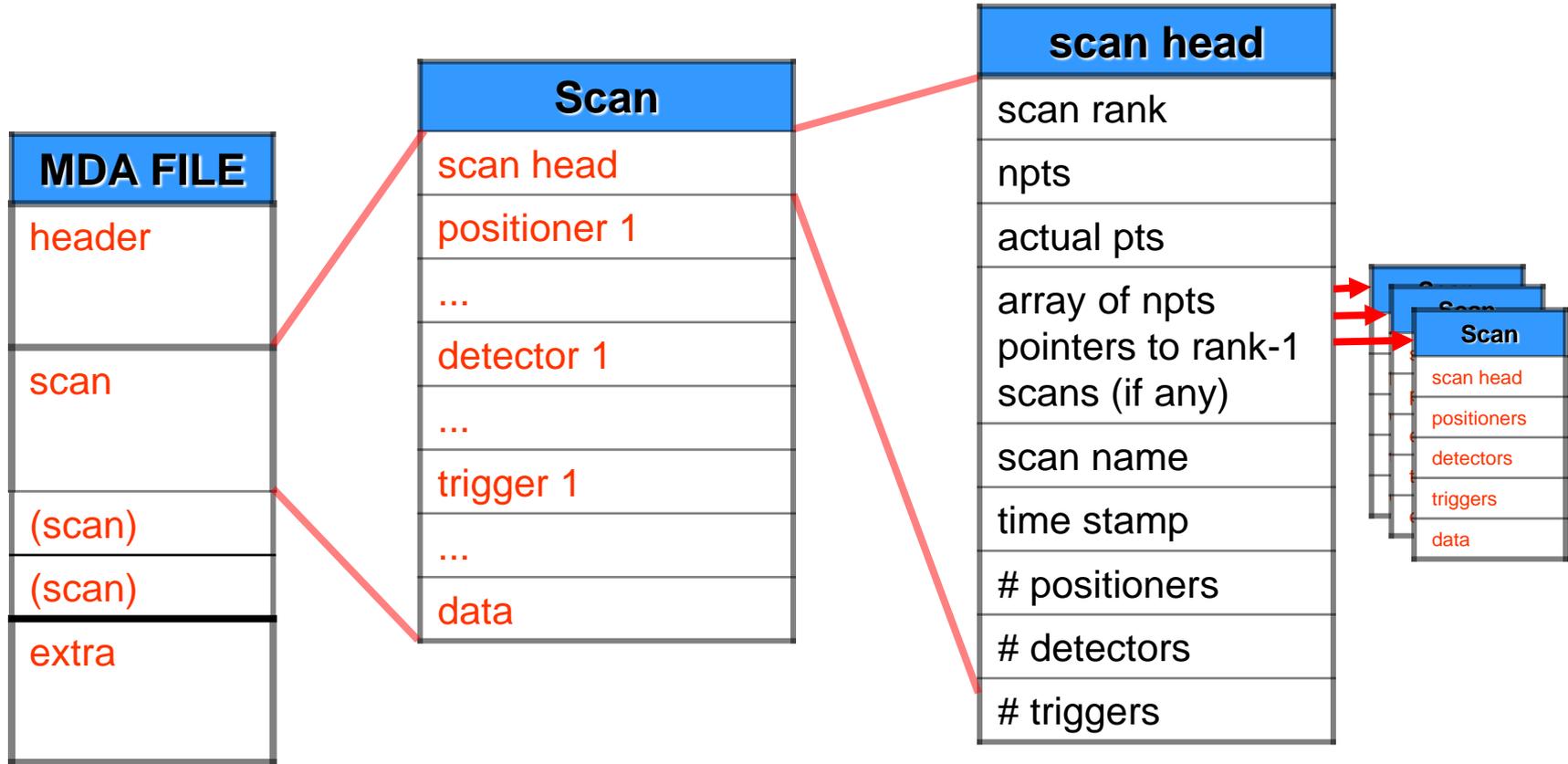
Description

If not supplied, .DESC field is used

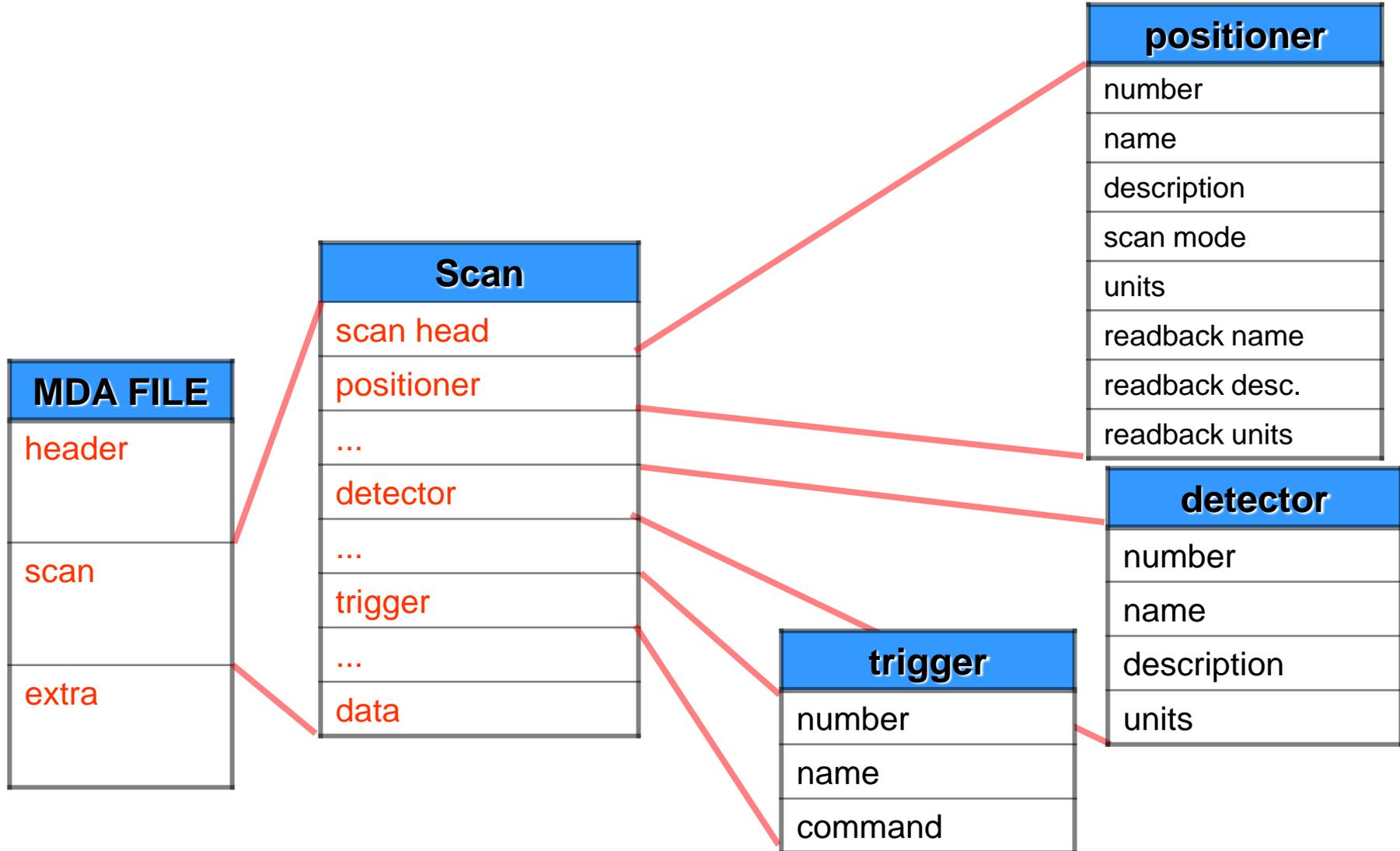
MDA file format



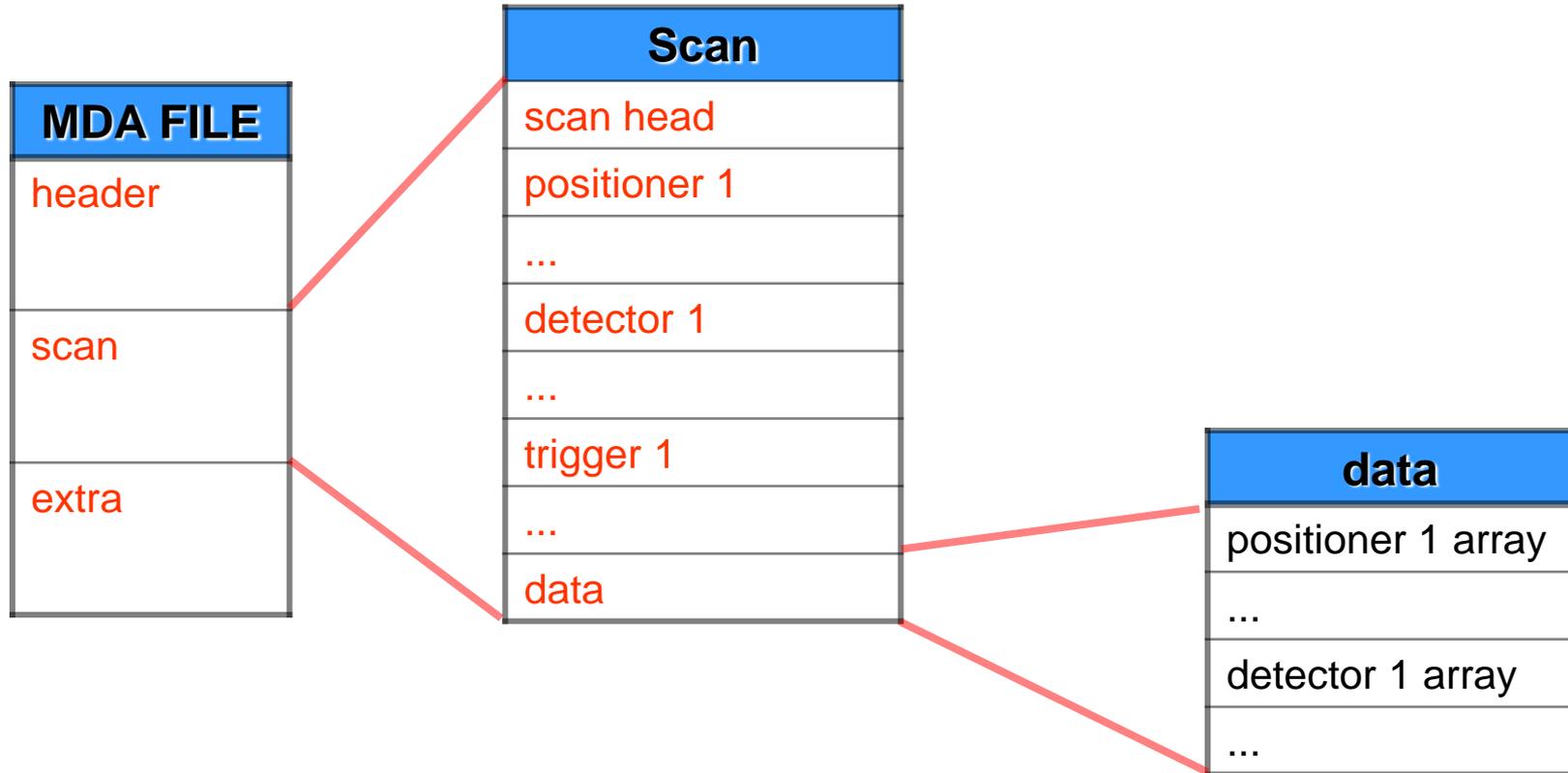
...MDA file format



...MDA file format



...MDA file format



Other data-acquisition-related software

- **Data-visualization tools for use with synApps**
 - scanSee
 - mdaExplorer (synApps utils)
 - dview
 - sview
 - webics <https://github.com/djvine/webics>

- **areaDetector**
 - Allows any EPICS CA client to drive data acquisition.
 - Support `ca_put_callback()`, as needed by the **sscan** record.

- **software to read, manipulate .mda files**
 - mda.py
 - mdautils

Completion reporting

- **Simple prescription for databases contained within a single ioc:**
 - Use only PP links and forward links in execution chain.

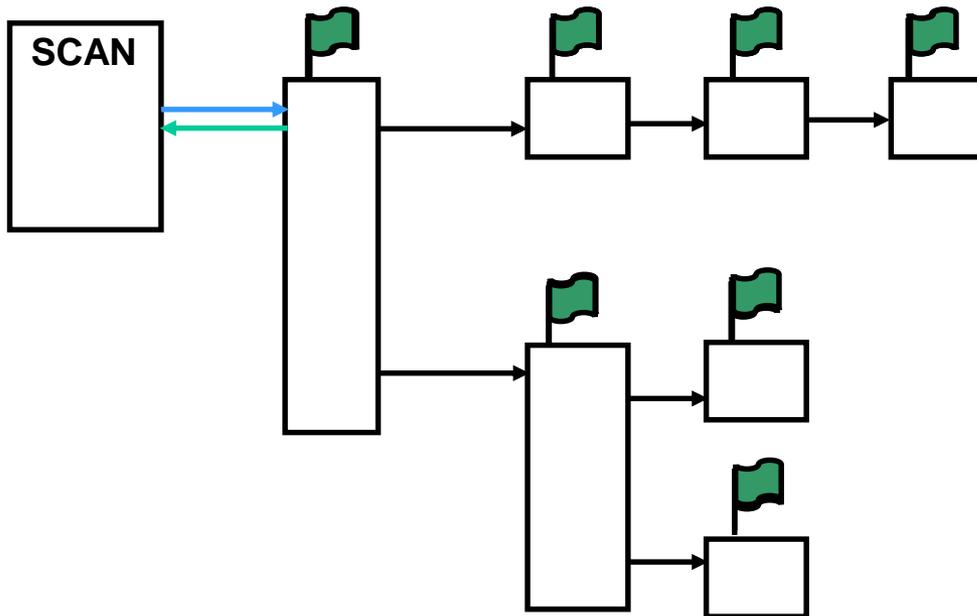
- **Database operations spanning more than one ioc:**
 - Use records with put_callback links to span iocs:
 - **calcout** with asynchronous device support
 - **sscan**, **swait** (i.e., a synApps “userCalc”)
 - **sseq** or **sCalcout** (with .WAIT* = “Wait”)

- **Cases in which a CA client performs part of the operation:**
 - 1) Database sets a **busy** record via PP or put_callback link.
 - 2) CA client clears the **busy** record when operation is done.

- **Cases in which part of the operation is driven by a CP link:**
 - Not different from above; a CP link is a CA client

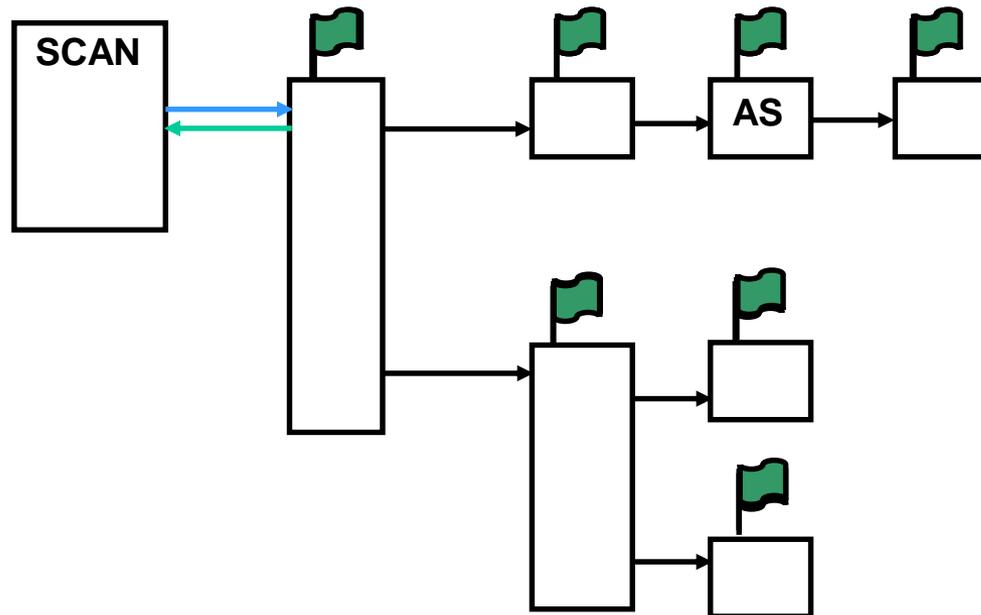
...Completion reporting

- Use only PP links and forward links in execution chain.



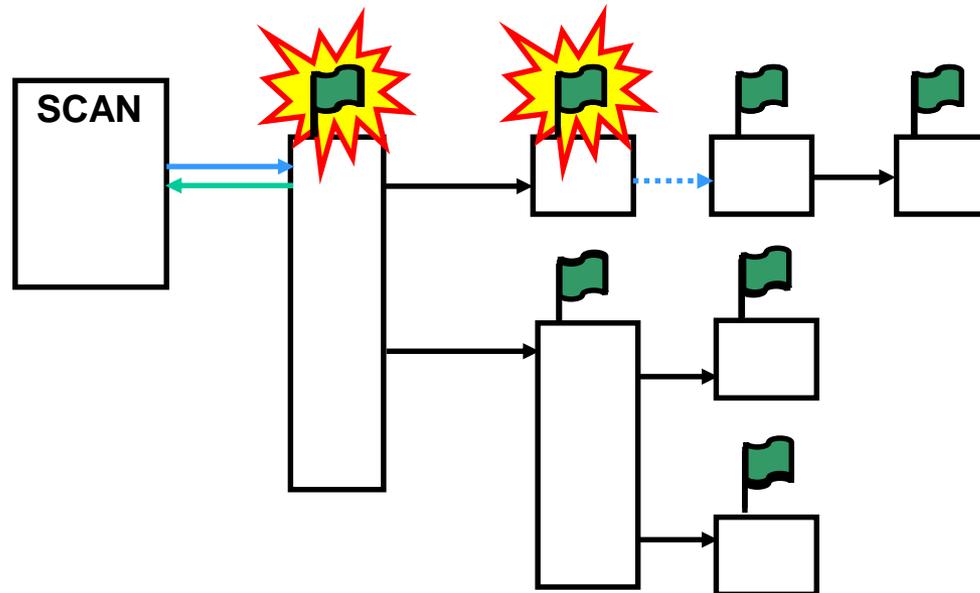
...Completion reporting

- Same as before, but with an *asynchronous* record



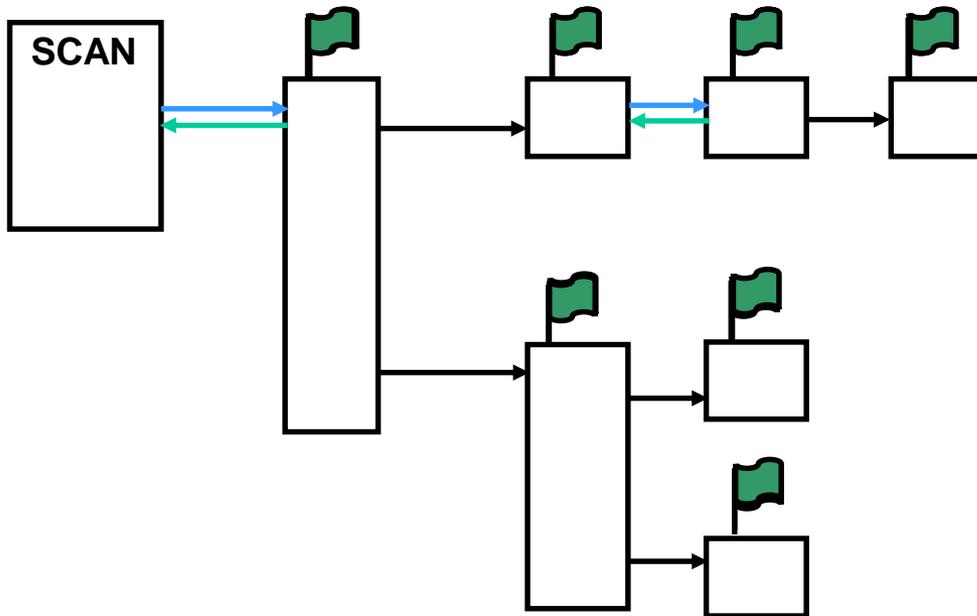
...Completion reporting

- Premature “DONE” report, because CA-link execution is not traced



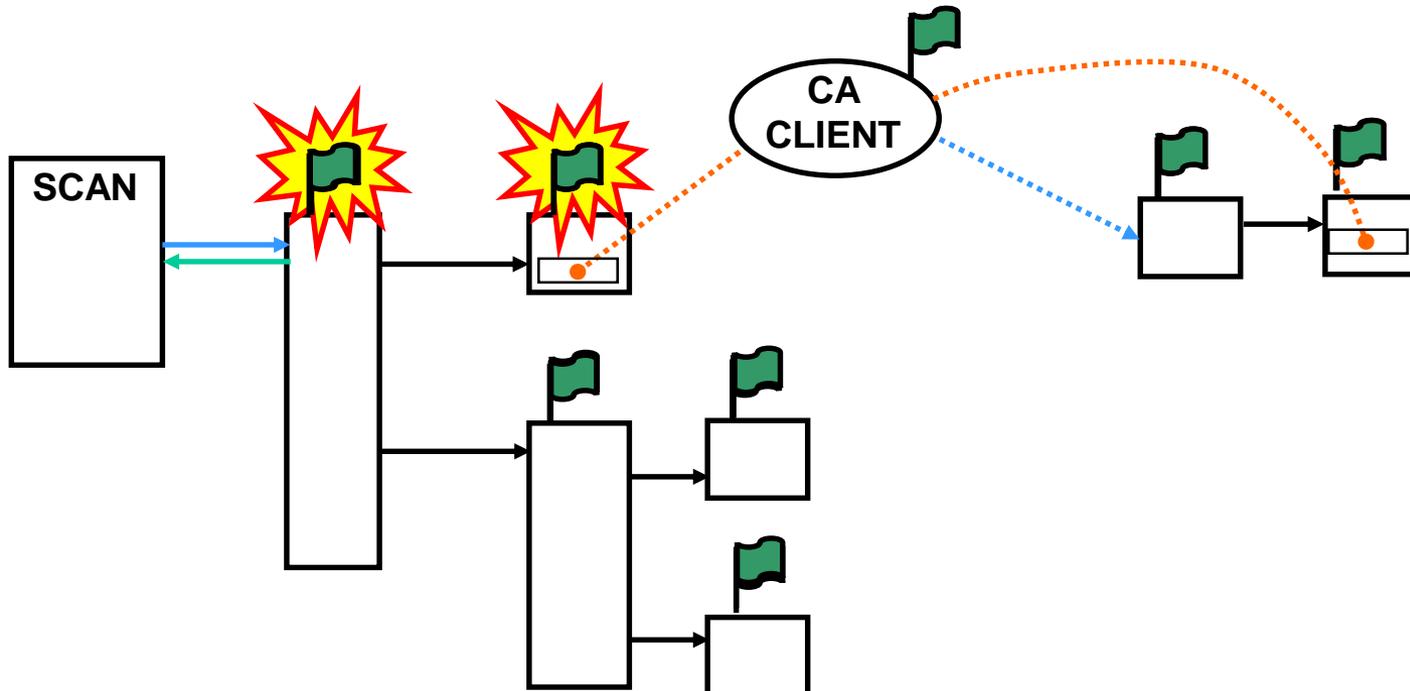
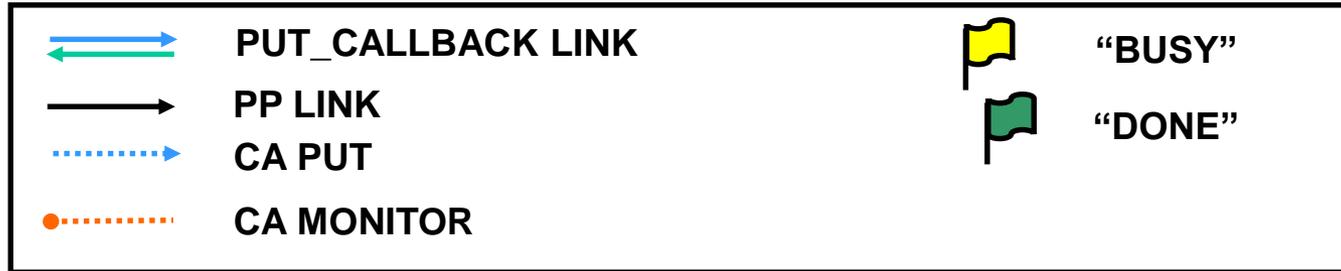
...Completion reporting

- Premature-DONE problem fixed with a PUT_CALLBACK link



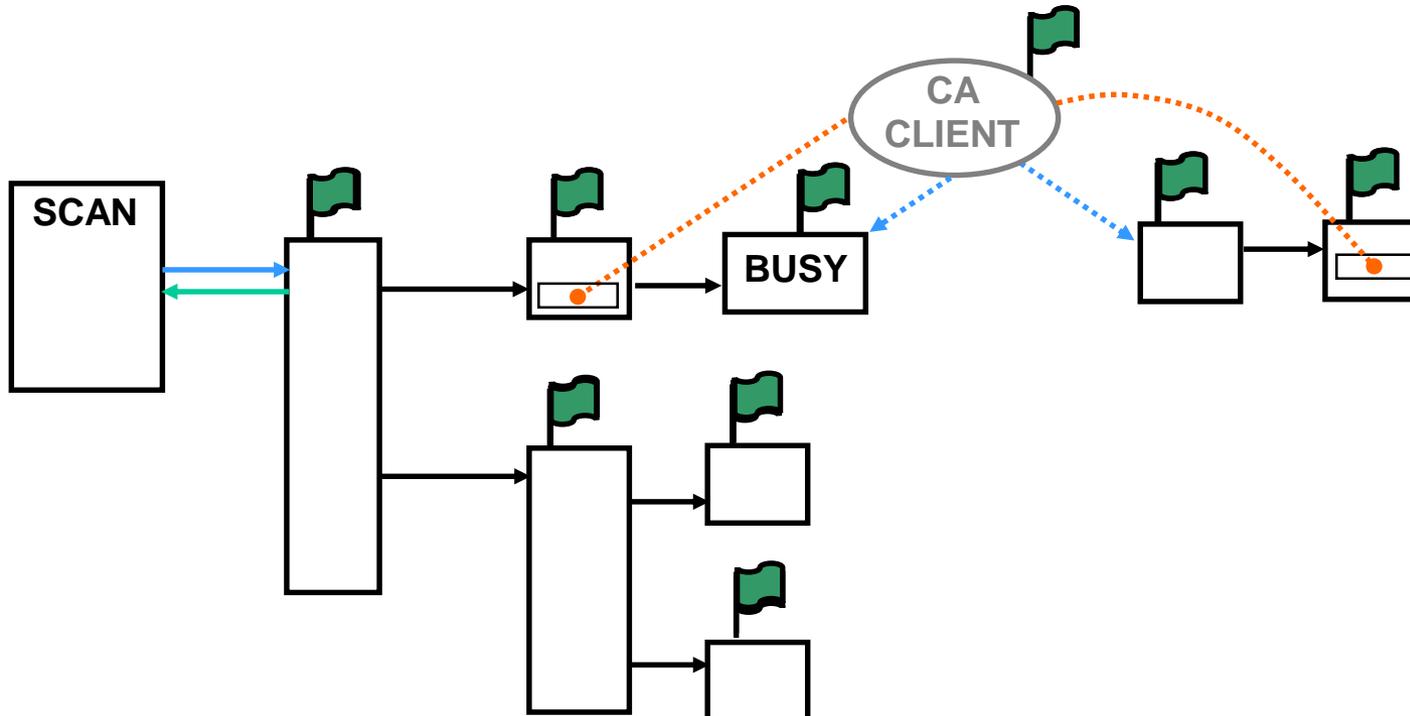
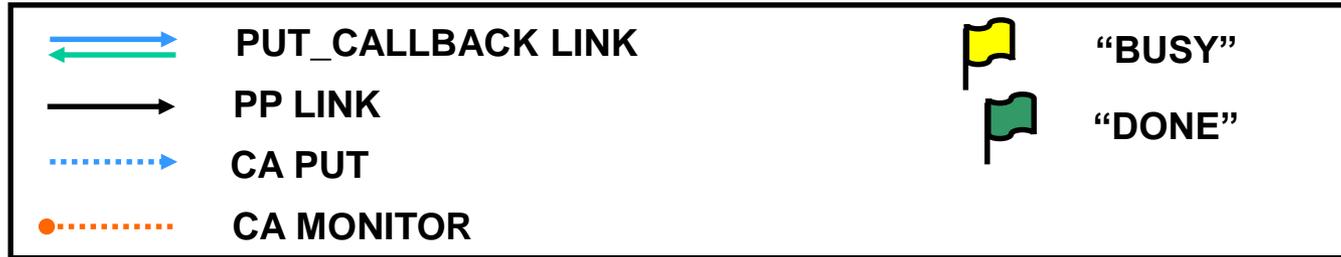
...Completion reporting

- Premature “DONE” because CA-client processing is not traced



...Completion reporting

- Premature “DONE” problem fixed with a ‘BUSY’ record



...Completion reporting

