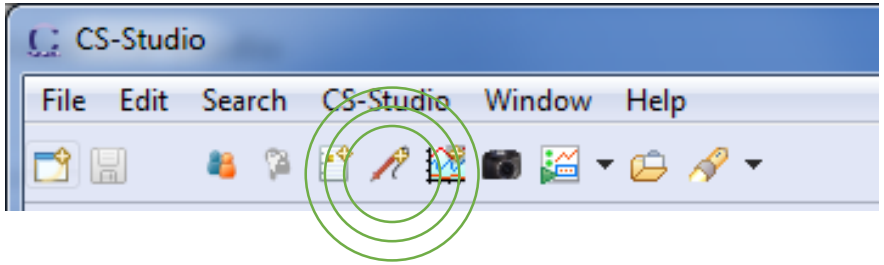
CS-Studio:
Probe

Probe

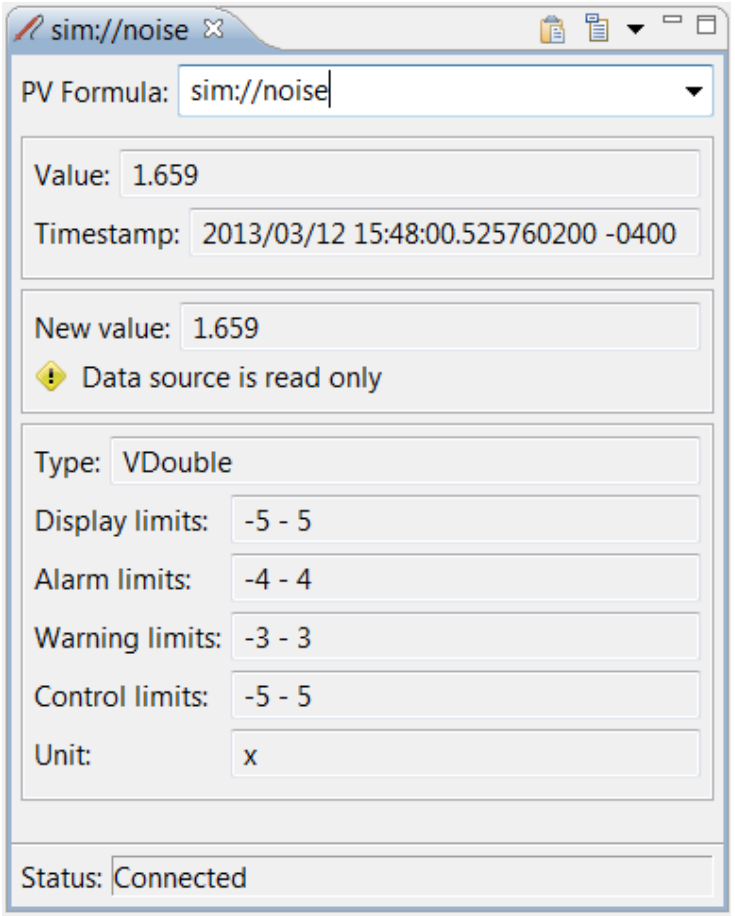
- A simple application to connect to pv's
  - Monitor pv's (caget, camonitor)
  - Write to pv's (caput)
- Opening probe
  - Menu:
    CSS → Diagnostic Tools → Probe
  - Toolbar:



  - Context Menu*:
    Process Variable → Probe

Probe - getting started

- Connect to a simple scalar pv (simulation)
  - In the PV Formula field type “sim://noise” and hit the return key
- Connect to a simple scalar pv (softIOC)
  - In the PV Formula field type “XF:31IDA-OP{Tbl-Ax:X1}Mtr.RBV” and hit the return key



Probe - deciphering the data

- Value:

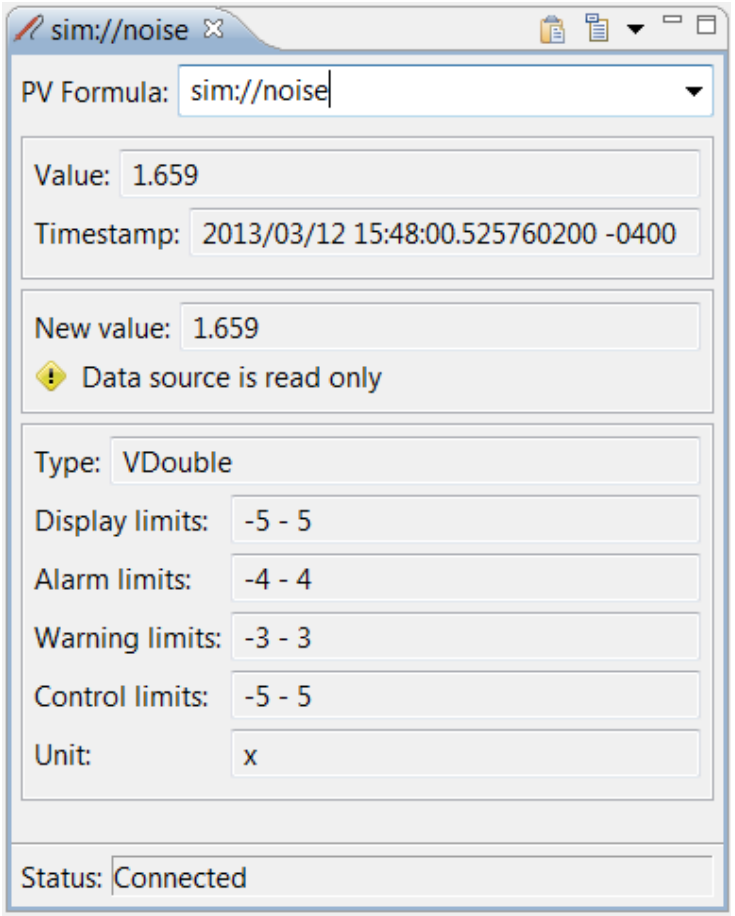
  Shows the current value of the pv and the timestamp associated with it

- Change Value:

  The new value field is used to write new values to the pv
  If the pv is read only this field is disabled

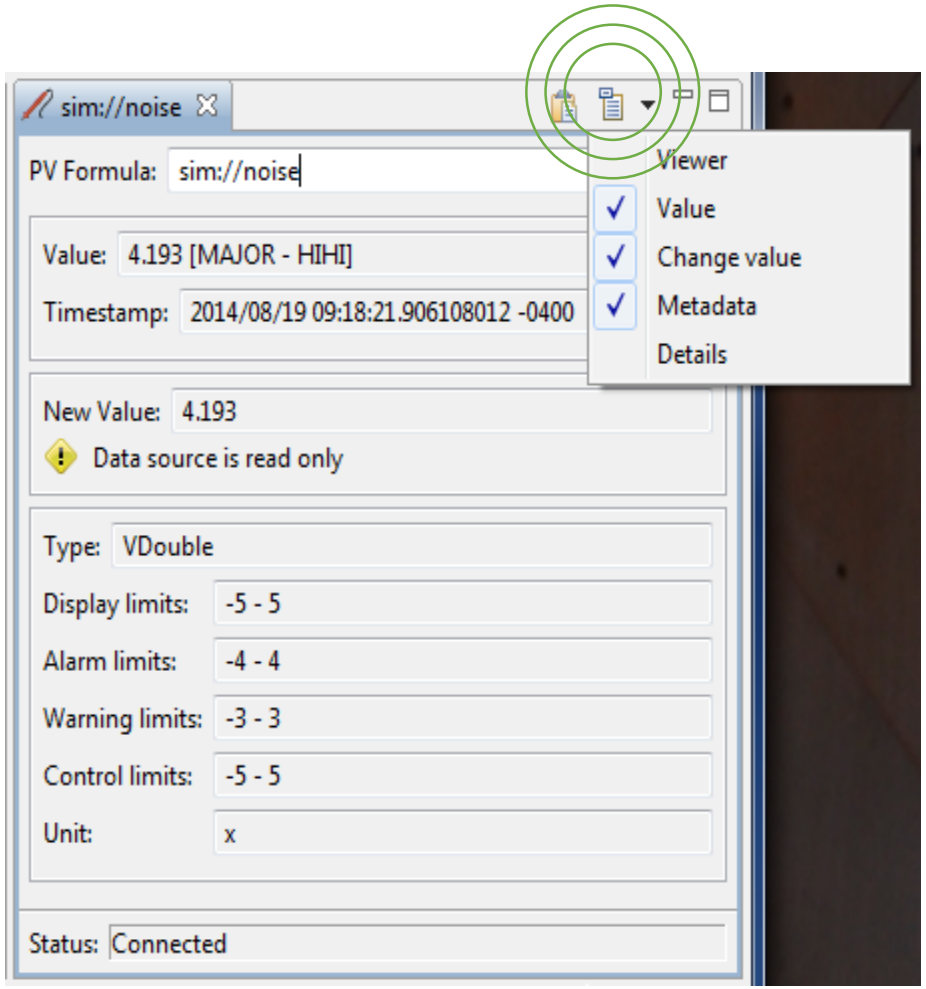
- MetaData:

  Additional information about the pv Type, Display limits, Alarm limits, units…..



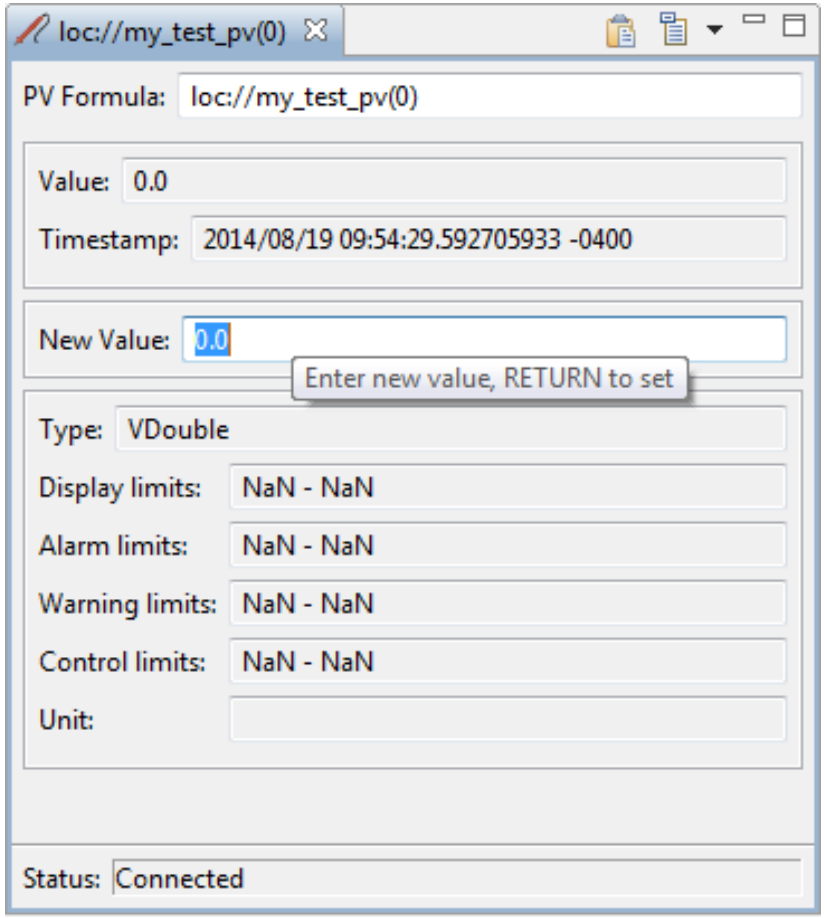
Probe - deciphering the data

- The drop down menu allows you to selected the additional information about the pv and the connection to be displayed
  - Viewer
  - Value
  - Change value
  - Details



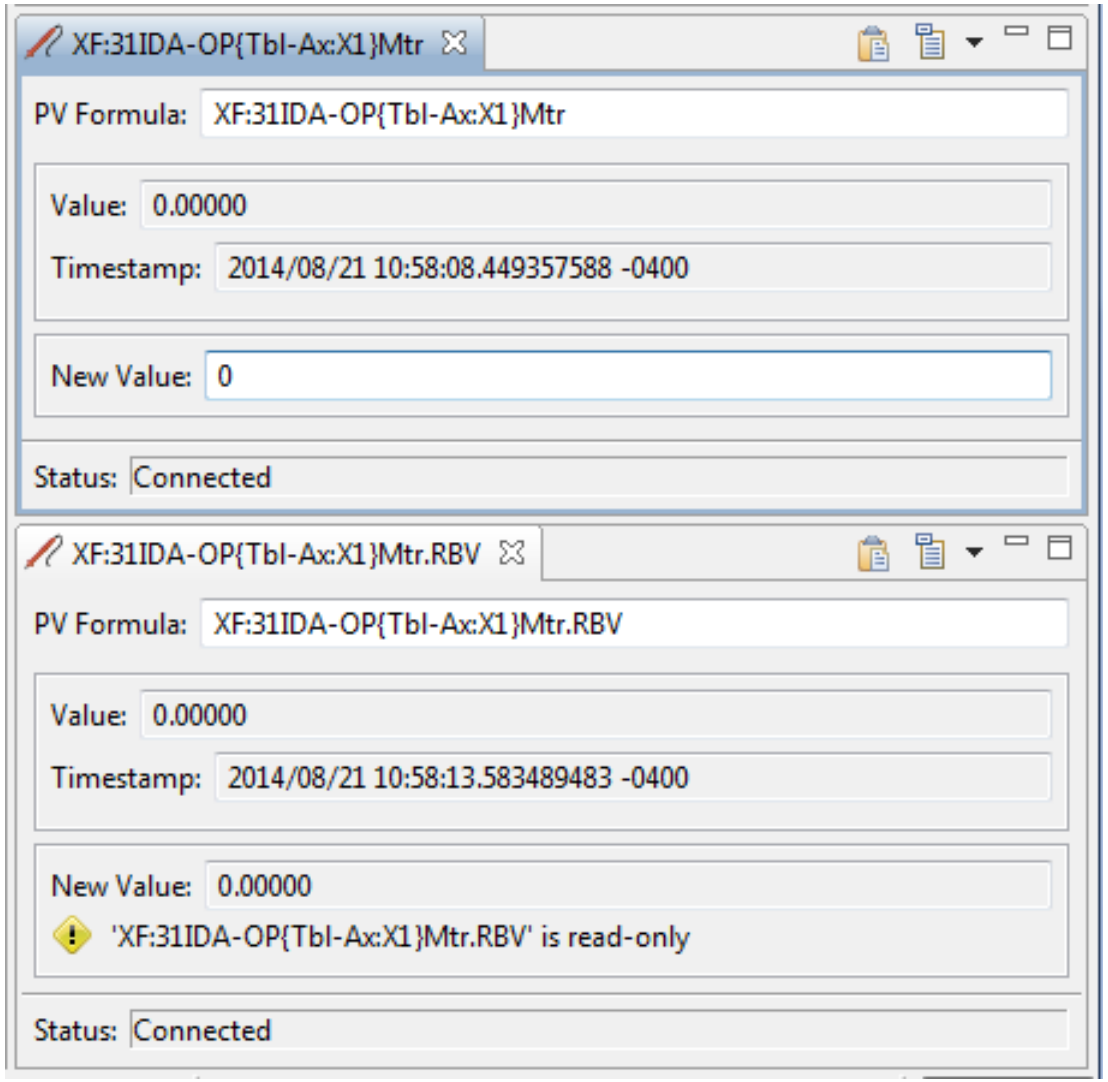
Probe - writing a value

- Connect to a simple scalar pv (local)
  - In the *PV Formula:* field type “loc://my_test_pv(0)” and hit RETURN
  - In the *New Value*: field enter any valid number and hit RETURN



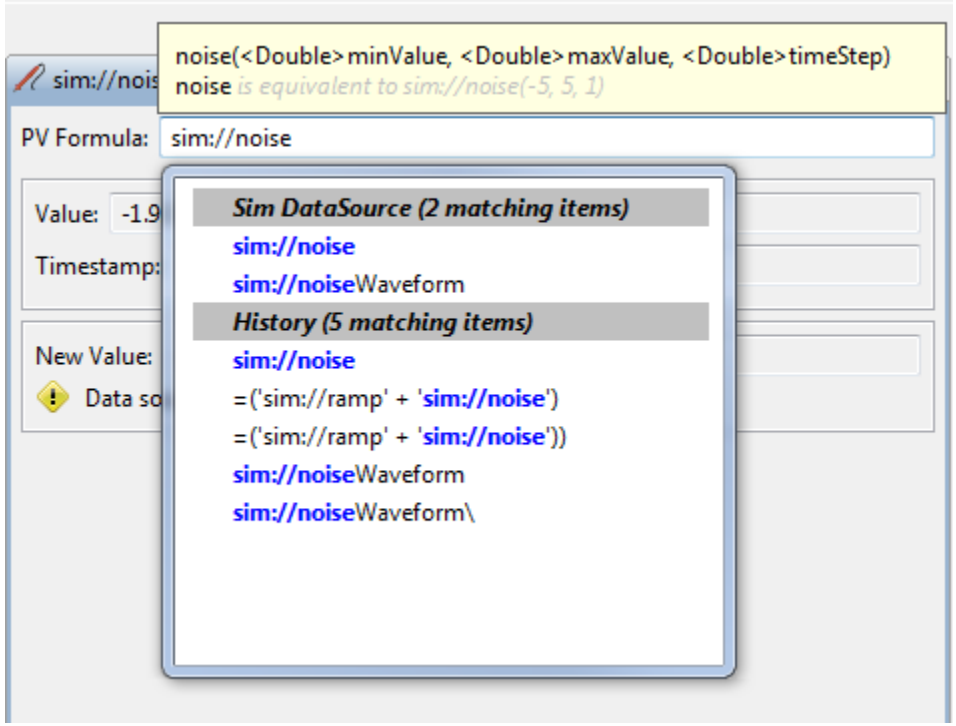
Probe - writing a value

- Open 2 probes
- Move the probe views so that they can be both viewed simultaneously
- In the first probe enter the setpoint pv
  PV: *XF:31IDA-OP{Tbl-Ax:X1}Mtr*
- In the second probe enter the Readback pv
  PV: *XF:31IDA-OP{Tbl-Ax:X1}Mtr.RBV*
- Set a new motor position (-100 and 100)



Probe - Autocomplete

- Autocomplete
  - History
  - ChannelFinder
  - Predefined simulation pv’s (Datasources)
  - Formula functions



Probe - Help

- Help → Help Content

  CSS Applications → Diagnostic Tools → Probe

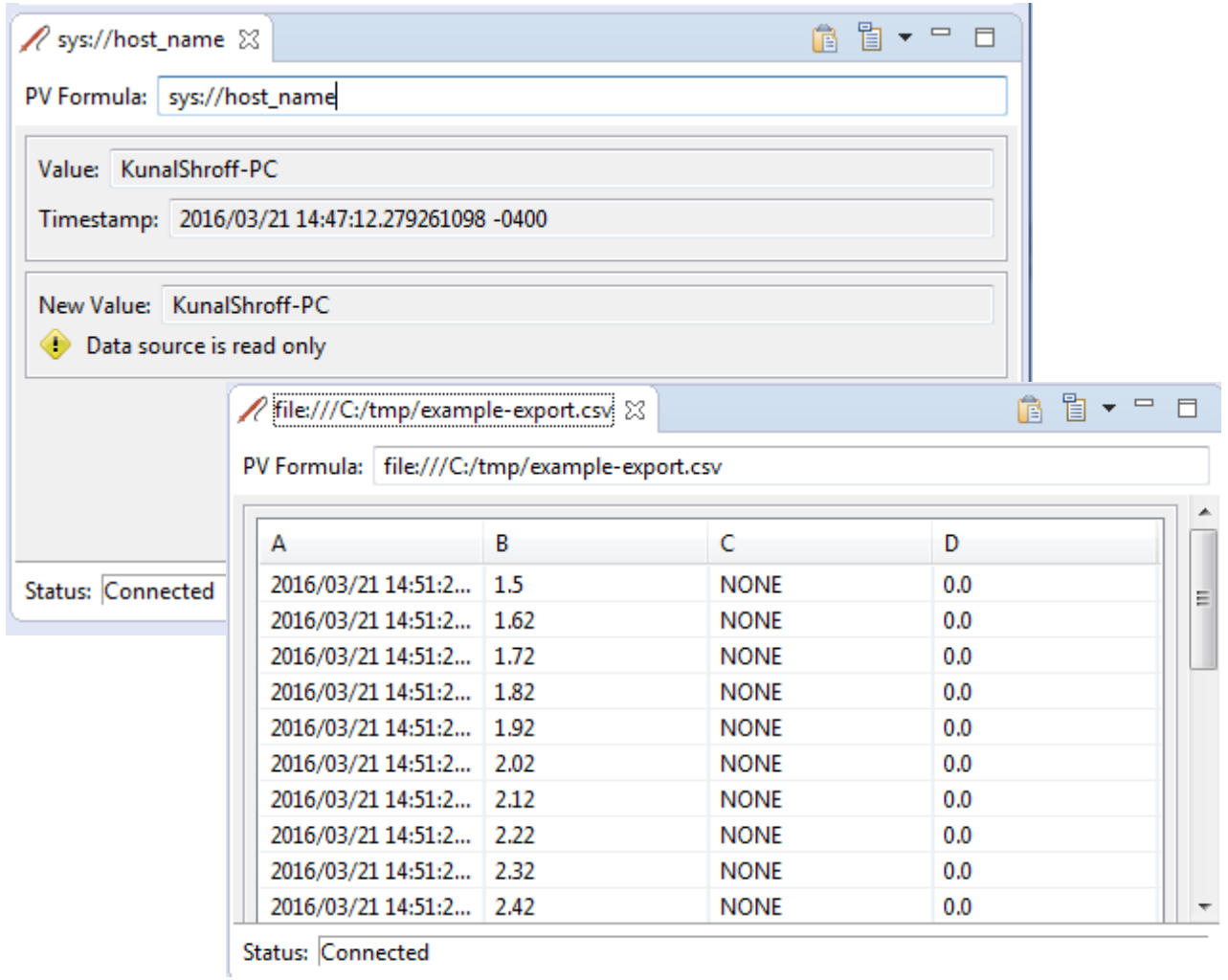
Probe – process variable (Datasource)

type://some_pv_name

- Datasources
  - System (*sys://*)
  - Local (*loc://*)
  - Simulation (*sim://*)
  - File(*file://*)
  - PVAccess (*pva://*)

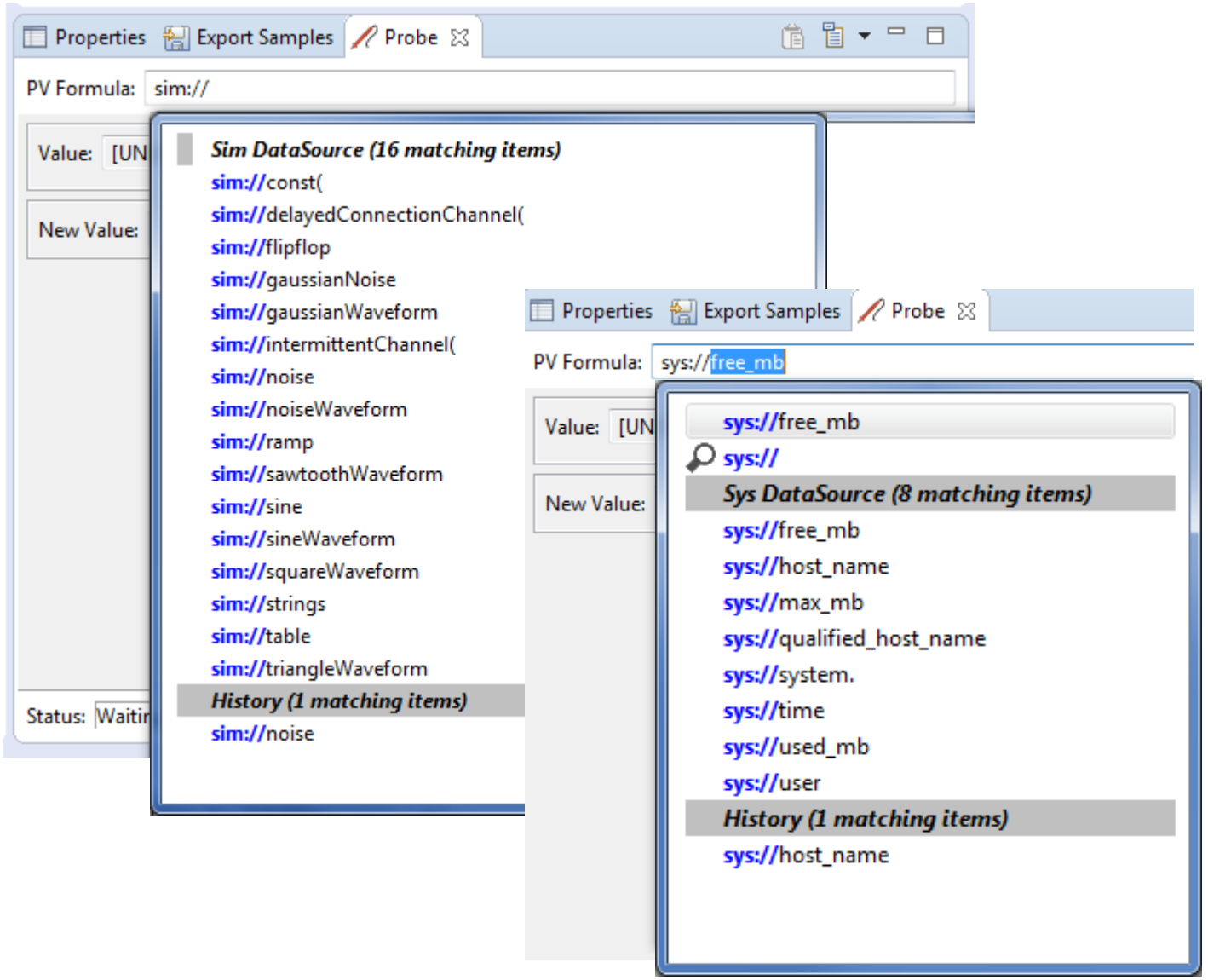
- Help

  CSS Core → Process Variables



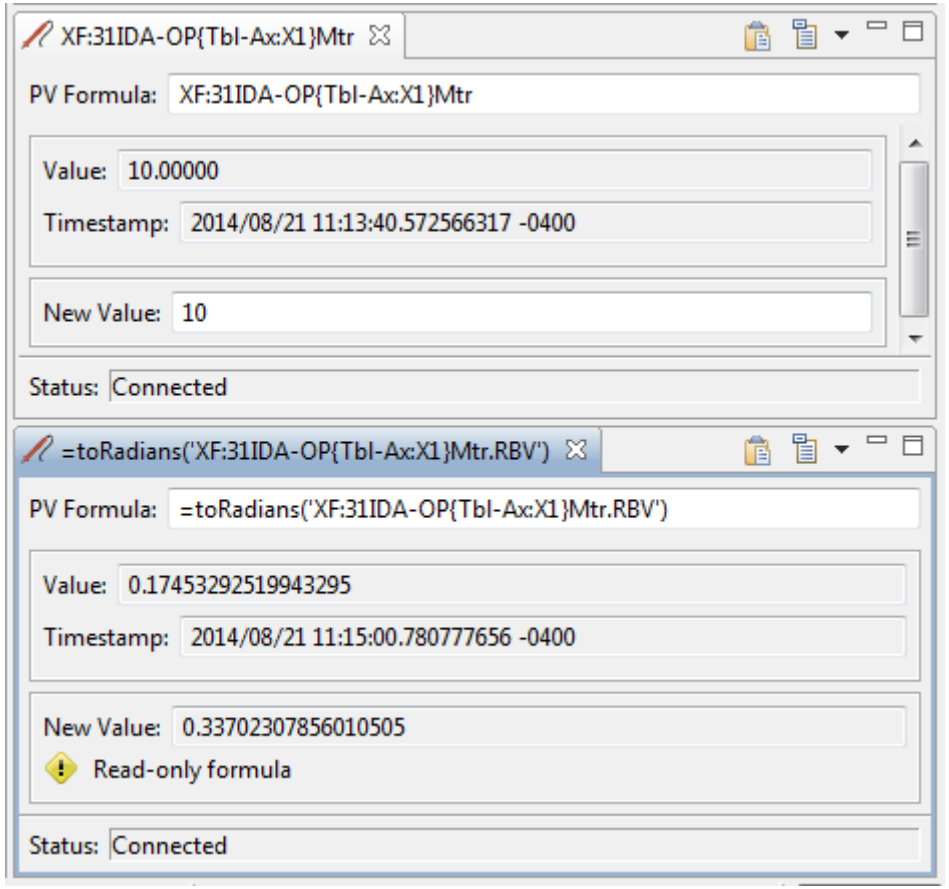
Probe - Autocomplete Datasources

- Autocomplete will list the various pv's available for a particular datasource



Probe - pv's and formulas

- Formulas
  - Mathematical operations
  - Logical operations
  - Array operations
  - Create/modify tables and arrays
- Start with a '='
- Don't use formulas for things that should be done in the IOC



Probe - pv's and formulas

CSS →

Debugging →

Formula Functions

Formula Functions

Name	Description
▷ array	
◢ math	
abs(VNumber arg): VNumber	Absolute value
acos(VNumber arg): VNumber	Arc cosine
asin(VNumber arg): VNumber	Arc sine
atan(VNumber arg): VNumber	Arc tangent
cbrt(VNumber arg): VNumber	Cubic root
ceil(VNumber arg): VNumber	Ceiling function
cos(VNumber arg): VNumber	Cosine
cosh(VNumber arg): VNumber	Hyperbolic cosine
exp(VNumber arg): VNumber	Exponential
floor(VNumber arg): VNumber	Floor function
integrate(VNumber value): VNumber	Integrates the given signal in time
log(VNumber arg): VNumber	Natural logarithm
log10(VNumber arg): VNumber	Base 10 logarithm
signum(VNumber arg): VNumber	Sign function
sin(VNumber arg): VNumber	Sine
sinh(VNumber arg): VNumber	Hyperbolic sine
sqrt(VNumber arg): VNumber	Square root
tan(VNumber arg): VNumber	Tangent
tanh(VNumber arg): VNumber	Hyperbolic tangent
toDegrees(VNumber arg): VNumber	Converts radians to degrees
toRadians(VNumber arg): VNumber	Conerts degrees to radians
▷ numericOperators	
▷ string	
▷ table	

Probe - Autocomplete formula

- Figure 1:
  Autocomplete will list the formula functions matching your text.
- Figure 2:
  Autocomplete will help you fill the formula function with the correct parameters.

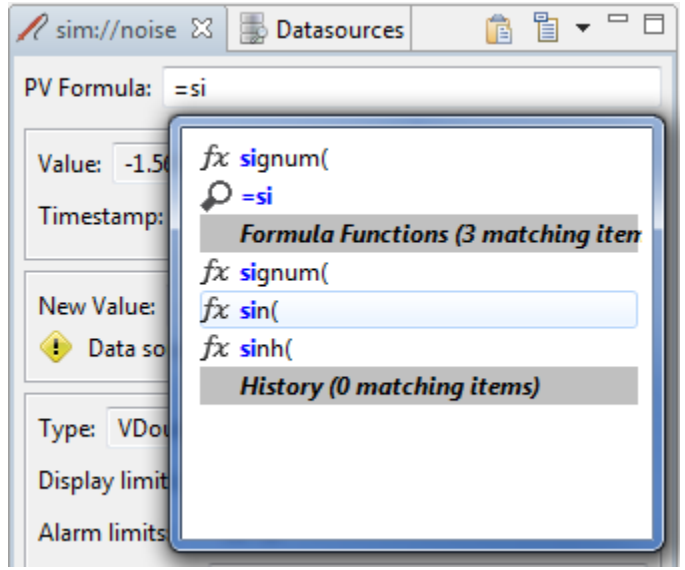


Figure 1

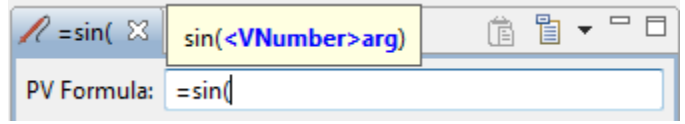


Figure 2