TINE Release 4.0 News

(Nov 11, 2011: That was the month that was !)

“What a long, strange trip it’s been ….”

Release 4.2.7

- Improvements in version 4.2.7
  - Stable against *port scans* ! (also 4.2.6)
    - 1 of the 3 TCP listening sockets caught in an infinite loop if first 2 bytes = 0 !
    - tested against all manner of corrupt input data.
  - Some new *meta properties*
    - ".URL", ".MAX", ".MIN", ".XMAX", ".XMIN"
    - ".EGU", ".XEGU" now WRITEABLE (but volatile)
    - New property description parsing
  - New Property signal: *PS_PROCESSED*
  - *State-Change* callback triggers
  - Establish a '*root*' context for decorated contexts.
  - Command line debugging: allow *negative filters*
  - Group Server: accept device name *pre-* and *post-fix* decorations

Release 4.2.7

- Property Description parsing:

  "[vscale=<min>:<max> <units>]
  [hscale=<xmin>:<xmax> <xunits>]
  [vplot=<plot style>]
  [hplot=<plot style>]
  [url=<url string>]
  [desc=<description>]"

Release 4.2.7

- Legacy style description (still parsed correctly)

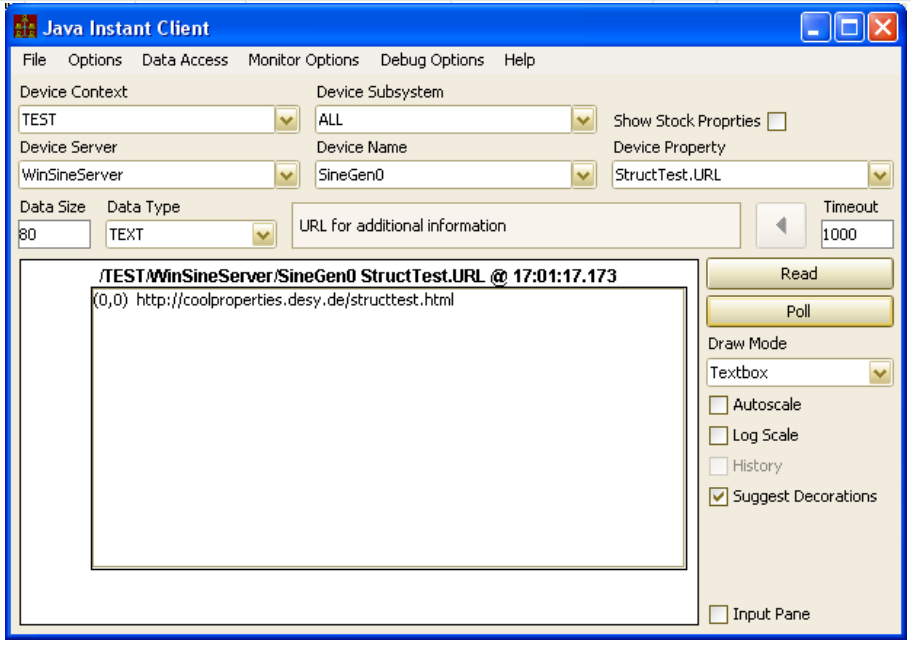
  "[<min:max> <units>]
  [<xmin:xmax> <units>]
  <description>"

- Many registered properties still do not supply this information !
  - => settings = 0 to 0
  - units = ""
  - description = ""

Release 4.2.7

- example export.csv:

EXPORT_N	LOCAL_NA	PROPERTY	PROPERTY	PROPERTY	ACCESS	FORMAT	NUM_	DESCRIPTION
WinSineS	SINEQM	Sine	8192	1	READ|XREAD	float.SPECTRUM	10	[vscale=-1000:1000 V][hscale=0:1000 ms][desc=Sine curve]
WinSineS	SINEQM	Amplitude	10	2	READ|WRITE	float.CHANNEL	10	[1:1000 V !LOG]Sine Curve Amplitude
WinSineS	SINEQM	Frequency	10	3	READ|WRITE|SAVE	float.CHANNEL	10	[1:60]Sine Curve Frequency
WinSineS	SINEQM	Phase	10	4	READ|STATIC	float.CHANNEL	10	[0:512]Sine Curve Phase
WinSineS	SINEQM	Noise	10	5	XREAD|WRITE	float.CHANNEL	10	[0:100 V]Sine Curve Noise Level
WinSineS	SINEQM	SineInfo	10	6	READ|WRITE	struct.SineInfo	10	Sine Generator Information
WinSineS	SINEQM	StructTest	10	8	READ|WRITE	struct.StCmp	10	[url=http://coolproperties.desy.de/structtest.html][desc=struct test]



Release 4.2.7

- Meta Properties
  - “.MAX”, “.MIN”, “.XMAX”, “.XMIN” duplicate information retrievable from “.EGU” and “.XEGU” with format CF_FLOAT or CF_USTRING
  - Now WRITEABLE
    - But changes are volatile !

Release 4.2.7

- New Property (Contract) Signal PS_PROCESSED !
  - sent after a contract has completely processed the returned data objects.
- Not true of PS_CALLED !
  - sent immediately after an *eqm dispatch* routine returns.
    - The returned data object could contain references which need to be evaluated.
- Note: PS_SENT sent later
  - and NOT guaranteed to be sent following every *eqm dispatch*!

Release 4.2.7

Signal functions …

```
int RegisterPropertySignalFunction ( const char *  eqm,
                                     const char *  prp,
                                     PRPSIG        fcn,
                                     int           mask,
                                     void *        ref
                                   )
```

Registers a property signal function.

If a property is accessed by remote callers, it will be represented within a server's contract list. The appropriate equipment module will be called, which is the extent of the transaction in most cases. The server can also optionally receive signals during the course of the transaction by attaching a signal function to the property in question. Signals will include

- **PS_ACCESS** (is being accessed by a new caller),
- **PS_RETRY** (is being retried),
- **PS_LATE** (is being returned late),
- **PS_PENDING** (is being called while last transmission still pending),
- **PS_CALLED** (has returned from call to the equipment module dispatch routine),
- **PS_PROCESSED** (has returned from dispatch call and has finished processing all returned information),
- **PS_SENT** (has been sent to caller),

Note:
the access bits CA_FIRST and CA_LAST can be used with the equipment module to check the scope of the caller's transaction.

Parameters:

eqm is the local equipment module name (maximum 6 characters in length) For example: "BPMEQM".

prp is the registered property for the signal function is to be applied.

fcn is the property signal function of prototype: void sigfcn(int signal,int contractId,int propertyId,int currentStatus,void *reference);

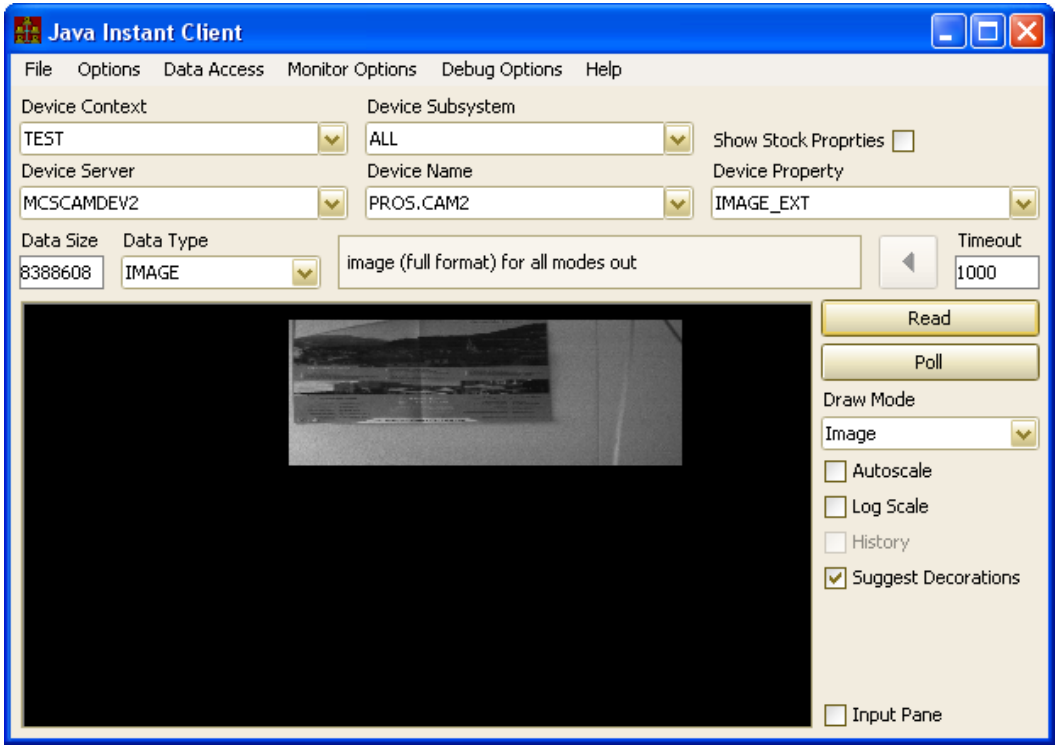
mask is a signal mask indicating which signals are of interest (use PS_ALL to receive all signals).

ref is a caller supplied reference which will be returned when the signal function. As this is a void pointer, it can refer to any structure or function the caller wants to have returned when the signal function is called.

Returns:
0 if successful, otherwise a TINE completion code

Release 4.2.7

- This all relates to the recent doocs2tine video bug !



Release 4.2.7

- State-Change callback triggers
  - Convenience functions !
  - C – API:

```
int RegisterStateChangeCallback ( SCCBFCNP   fcn,
                                  const char * eqm,
                                  const char * stateKey,
                                  void *       reference
                                )
```

Registers a state change callback dispatch function.

If STATE and GLOBALS servers are running, a server can receive state change notifications and react accordingly by calling this routine and passing a callback dispatch routine. The callback routine will be called only upon change of state. In the event of any i/o errors which prevent the server from receiving the state information, the state will be changed automatically to 'unavailable' following 5 consecutive readback errors.

Parameters:

fcn	is a reference to the state change dispatch routine to be called following a change of state This must have the prototype: void (*fcn)(const char *previousState,const char *currentState,void *reference). If a NULL is passed for this parameter then any callback routine will be de-registered.
eqm	the local equipment module name of the desired central server (e.g. "BPMEQM")
stateKey	is an optional specification of the desired state keyword from the GLOBALS server. If a NULL is passed, then the default of /<context>/GLOBALS/DeclaredState will be used.
reference	is a caller supplied void pointer which will be returned to the called in the dispatch routine. A NULL can be passed if no reference is required.

Returns:

0 if successful, otherwise a TINE completion code

Release 4.2.7

- e.g.

previous state

next state

```
enum operationModes
{
  mode_not_running,
  mode_electrons,
  mode_positrons
};

void myStaChg(const char *prv,const char *nxt,void *ref)
{
  printf("changed from %s to %s\n",prv,nxt);
  if (!strncmp(nxt,"running_e-"))
  {
    opMode = mode_electrons;
  }
  else if (!strncmp(nxt,"running_e+"))
  {
    opMode = mode_positrons;
  }
  else
  {
    opMode = mode_not_running;
  }
}

void PostSystemInit(void)
{
  // register equipment module(s) ...
  RegisterEquipmentModule("WinSineServer","SINEQM",NUM_DEVICES,sineqm,sininit,sinbkg,100,NULL);
  // other code omitted ...

}

void sininit(void)
{
  int cc = 0;
  // call restration routines ...
  registerSineProperties();
  registerSineDevices();

  // add a state change callback to the equipment module
  // use the default State Variable ...

  if ((cc=RegisterStateChangeCallback(myStaChg,"SINEQM",NULL,NULL)) != 0)
  {
    printf("could not register state change callback : error %d\n",cc);
  }
```

Use default state “key”

Release 4.2.7

- State-Change callback triggers (Java API):

void de.desy.tine.server.equipment.TEquipmentModule.setStateChangeTrigger(TStateChangeTrigger stateChangeTrigger, String stateChangeKey)

Establishes a state-change trigger callback for this equipment module. If a GLOBLAS server is running and providing state information, then any detected state change will be passed to the callback function provided. The callback must implement the TStateChangeTrigger class. It will receive both the current state and previous state (as Strings).

Parameters:

stateChangeTrigger is a reference to the callback instance.
stateChangeKey is the GLOBALS keyword providing the declared state for the context associated with this equipment module. If 'null' then the default keyword "DeclaredState" is assumed.

No "key" given => use default ("DeclaredState" in my context)

e.g.

```
    sineEqpModule.setStateChangeTrigger(testChange);
    cosineEqpModule.setStateChangeTrigger(testChange2);
  }
  TStateChangeTrigger testChange = new TStateChangeTrigger()
  {
    @Override
    public void update(String previousState, String thisState)
    {
      System.out.println("state changed from "+previousState+" to "+thisState);
    }
  };
  TStateChangeTrigger testChange2 = new TStateChangeTrigger()
  {
```

Release 4.2.7

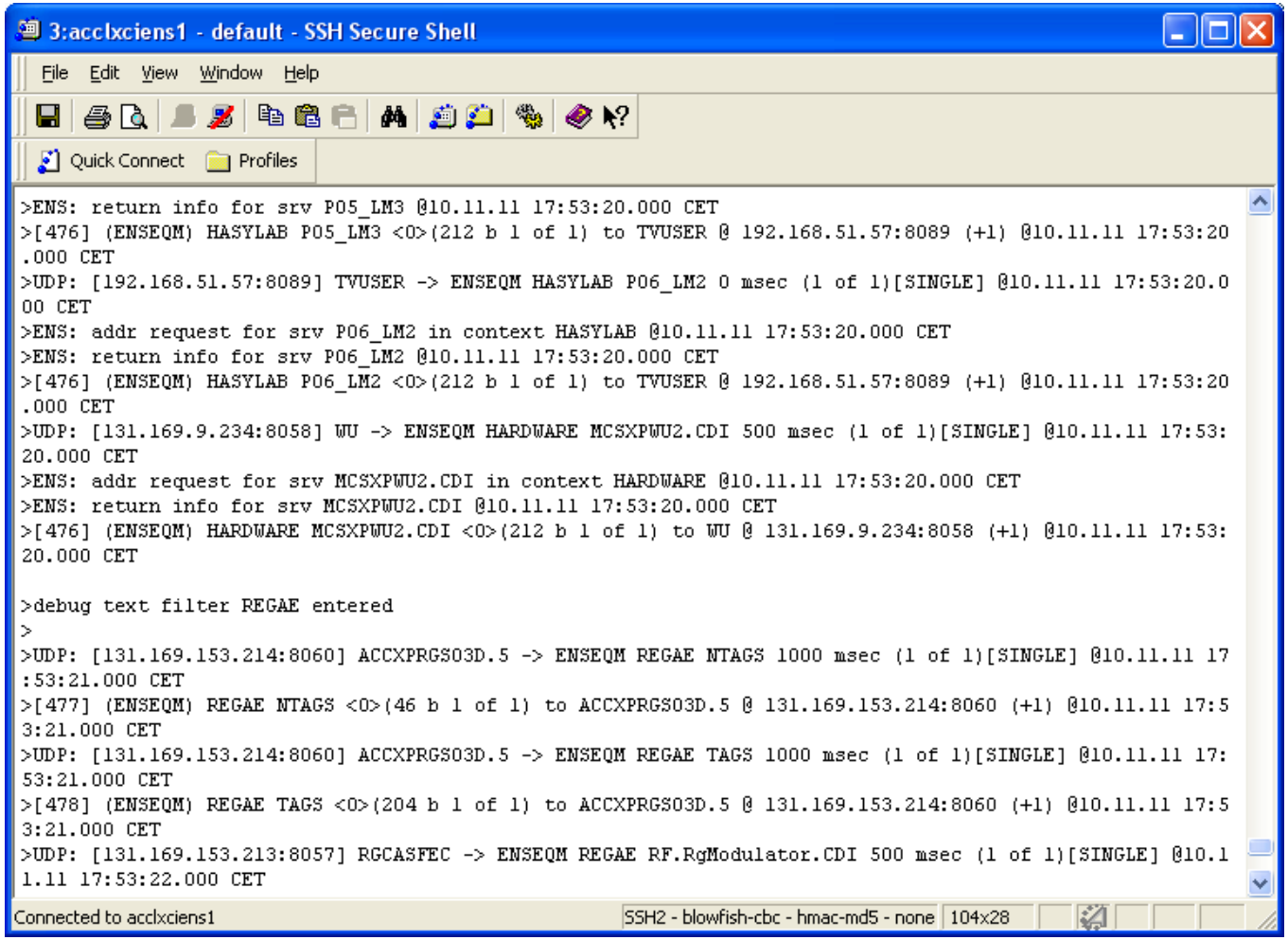
- '*root*' context used for accessing CAS and STATE servers.
  - remove decoration *if* subsystem is registered !
  - e.g. context "TTF2.RF" signals CAS in context "TTF2" if subsystem is registered.

Release 4.2.7

- ‘negative’ filters with command line debugging !

  e.g. the ENS with debug=1

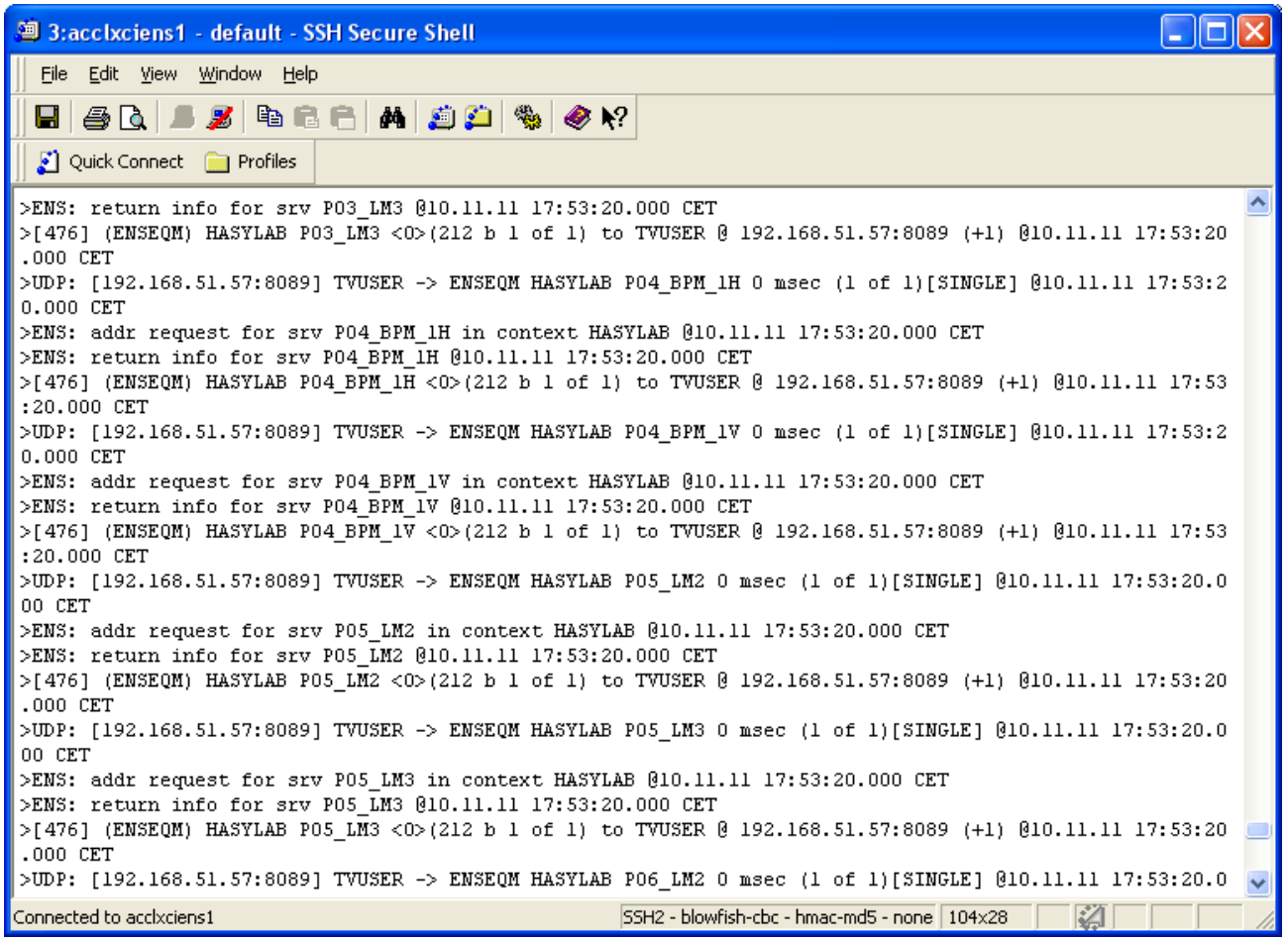
  a ‘positive’ filter:



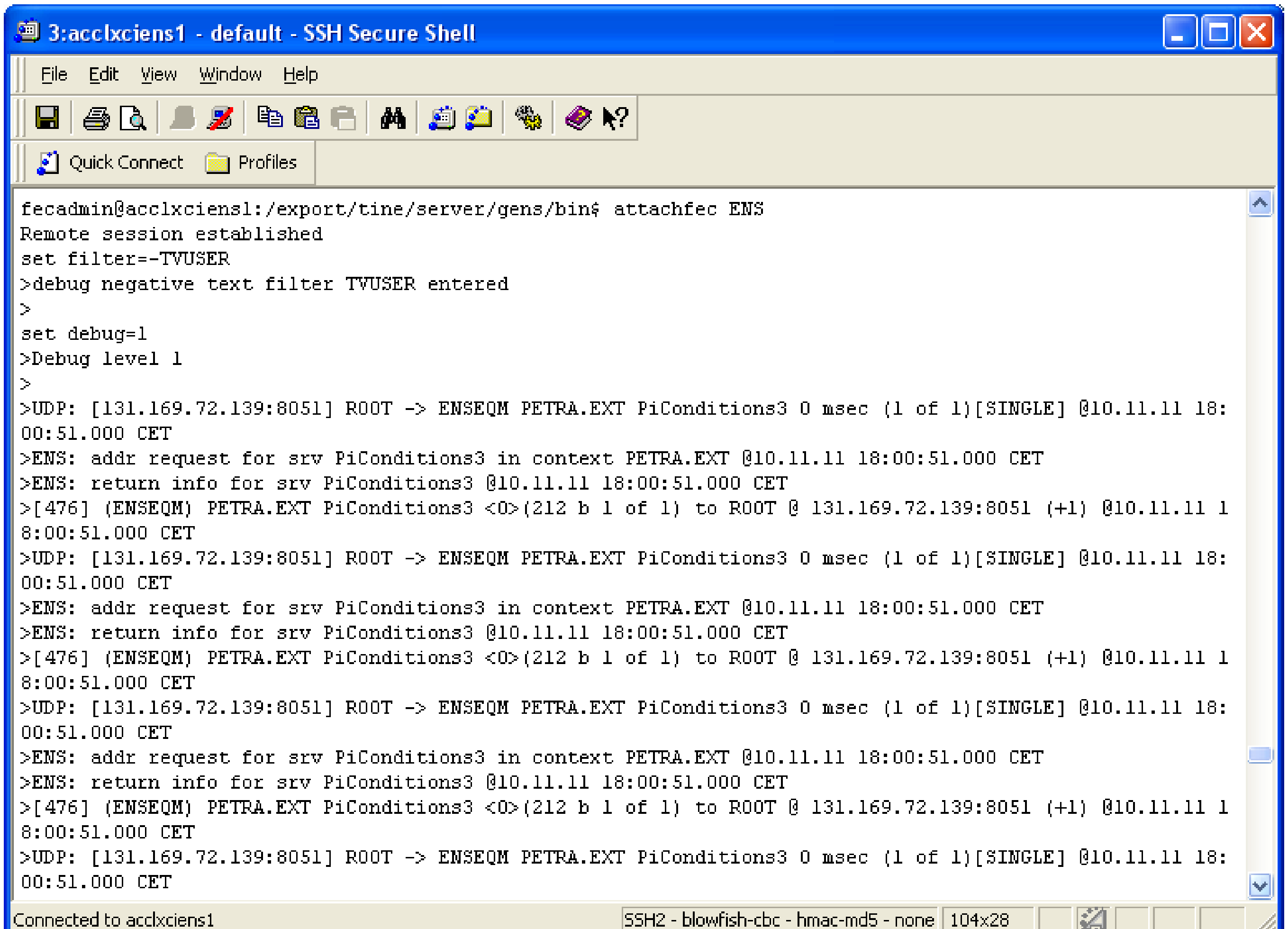
Release 4.2.7

But sometimes you want to filter ‘out’ and not filter ‘in’ :

e.g. get rid of “TVUSER” from the output !



Release 4.2.7


3:acclxciens1 - default - SSH Secure Shell
File Edit View Window Help
Quick Connect
Profiles
fecadmin@acclxciens1:/export/tine/server/gens/bin$ attachfec ENS
Remote session established
set filter=-TVUSER
>debug negative text filter TVUSER entered
>
set debug=1
>Debug level 1
>
>UDP: [131.169.72.139:8051] ROOT -> ENSEQM PETRA.EXT PiConditions3 0 msec (1 of 1)[SINGLE] @10.11.11 18:00:51.000 CET
>ENS: addr request for srv PiConditions3 in context PETRA.EXT @10.11.11 18:00:51.000 CET
>ENS: return info for srv PiConditions3 @10.11.11 18:00:51.000 CET
>[476] (ENSEQM) PETRA.EXT PiConditions3 <0>(212 b 1 of 1) to ROOT @ 131.169.72.139:8051 (+1) @10.11.11 18:00:51.000 CET
>UDP: [131.169.72.139:8051] ROOT -> ENSEQM PETRA.EXT PiConditions3 0 msec (1 of 1)[SINGLE] @10.11.11 18:00:51.000 CET
>ENS: addr request for srv PiConditions3 in context PETRA.EXT @10.11.11 18:00:51.000 CET
>ENS: return info for srv PiConditions3 @10.11.11 18:00:51.000 CET
>[476] (ENSEQM) PETRA.EXT PiConditions3 <0>(212 b 1 of 1) to ROOT @ 131.169.72.139:8051 (+1) @10.11.11 18:00:51.000 CET
>UDP: [131.169.72.139:8051] ROOT -> ENSEQM PETRA.EXT PiConditions3 0 msec (1 of 1)[SINGLE] @10.11.11 18:00:51.000 CET
>ENS: addr request for srv PiConditions3 in context PETRA.EXT @10.11.11 18:00:51.000 CET
>ENS: return info for srv PiConditions3 @10.11.11 18:00:51.000 CET
>[476] (ENSEQM) PETRA.EXT PiConditions3 <0>(212 b 1 of 1) to ROOT @ 131.169.72.139:8051 (+1) @10.11.11 18:00:51.000 CET
>UDP: [131.169.72.139:8051] ROOT -> ENSEQM PETRA.EXT PiConditions3 0 msec (1 of 1)[SINGLE] @10.11.11 18:00:51.000 CET
Connected to acclxciens1
SSH2 - blowfish-cbc - hmac-md5 - none
104x28

Release 4.2.7

- Group Server + device pre-/post-fix
  - ‘identical’ device servers (with identical device name lists) can establish a device group
  - distinguish among the various device lists by supplying a device prefix and/or a device postfix !

Release 4.2.7

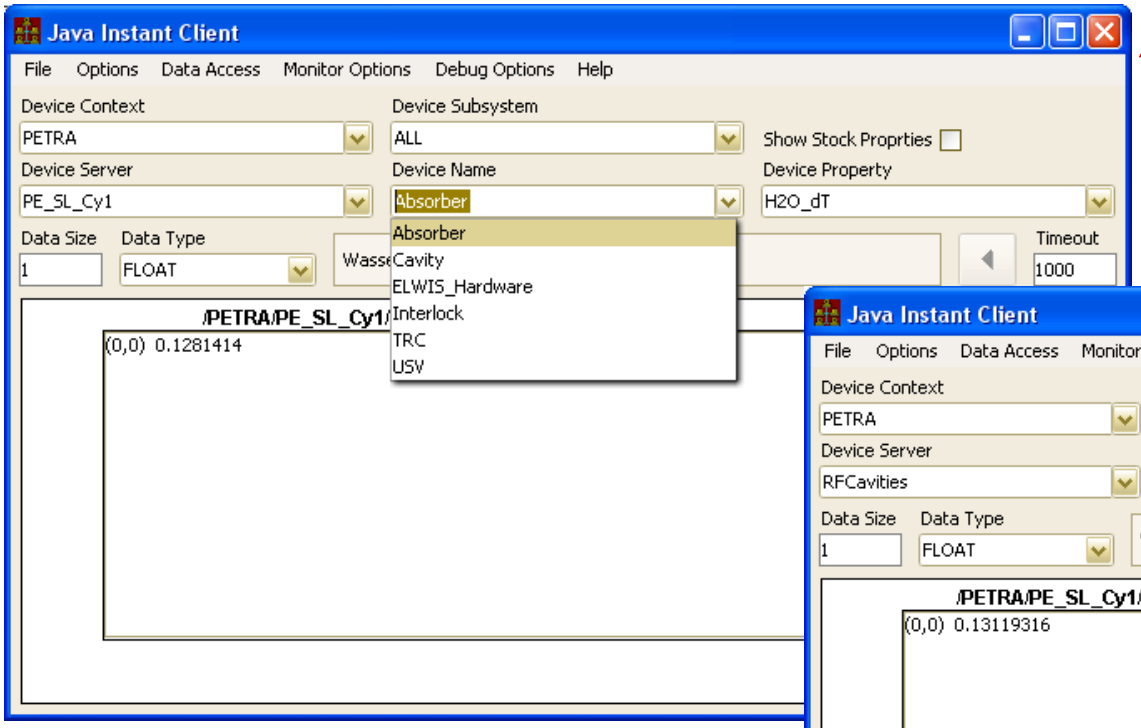
- Database configuration files
  - “GROUP_DEVICE_PREFIX” and “GROUP_DEVICE_POSTFIX”
    - optional columns in exports.csv or
    - optional tags in fec.xml
- API calls
  - C: JoinEquipmentGroupEx() ←
  - Java: setGroupDevicePrefix(), setGroupDevicePostfix()

Takes prefix and postfix tags

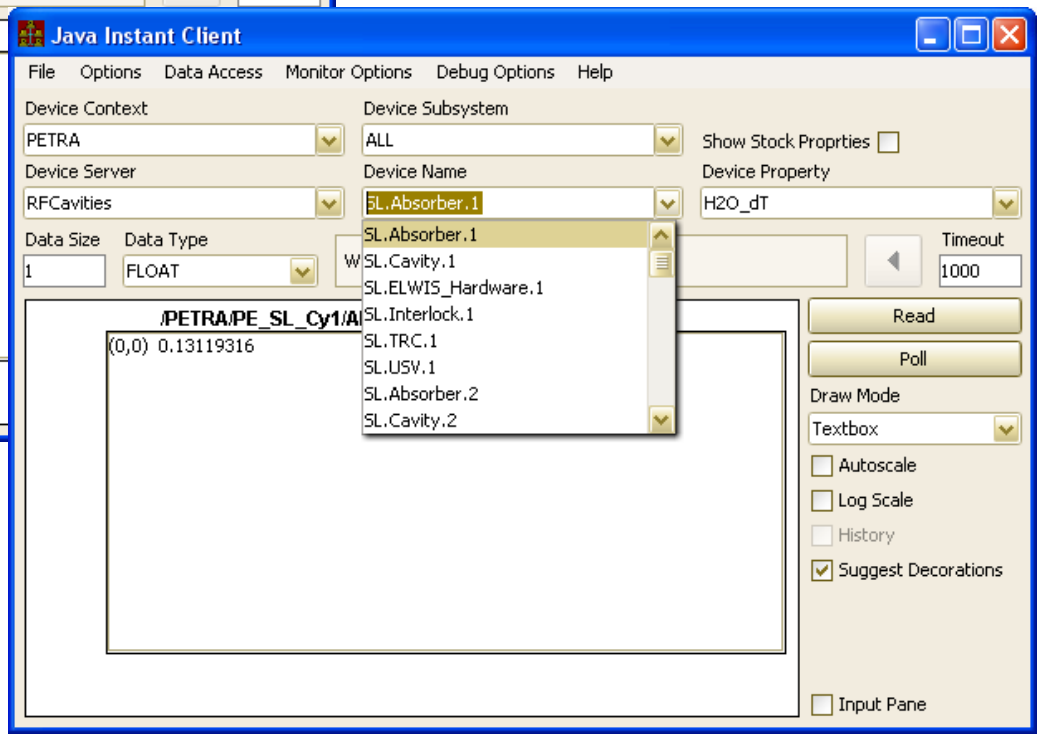
Release 4.2.7

12 Elwis Servers with same device list !

Use prefix “SL.” and “SR.” Postfix “.1” -> “.6” !



1 “logical” Server RFCavities :



Central Services

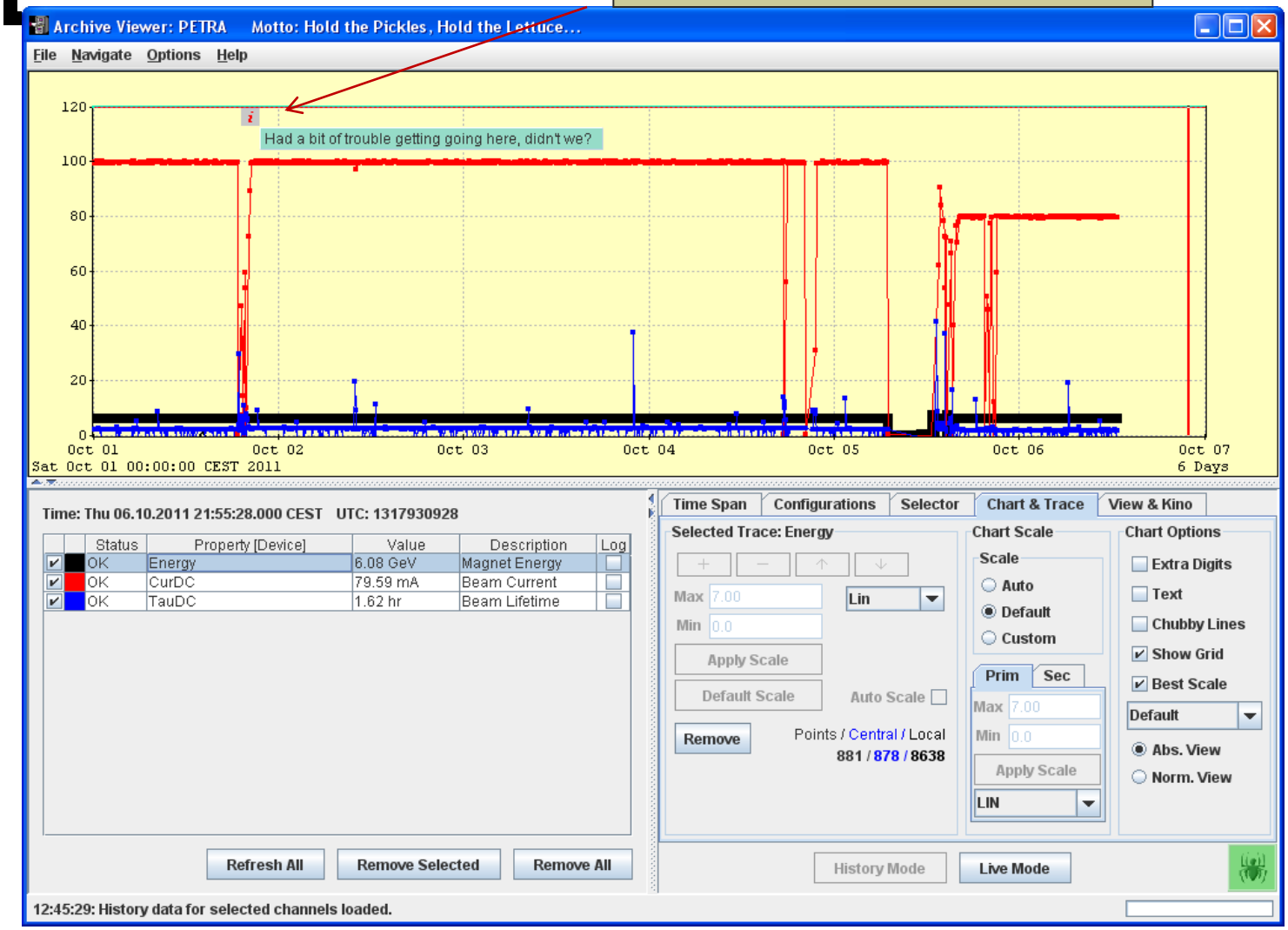
- Archive Viewer
  - annotations
  - status
  - add history records (local history)
- Archive Database Manager
  - becoming useable
- ENS Administration
  - can toggle ENS
- Alarm Viewer
  - add/edit alarm watch
  - edit alarm definition

Central Services

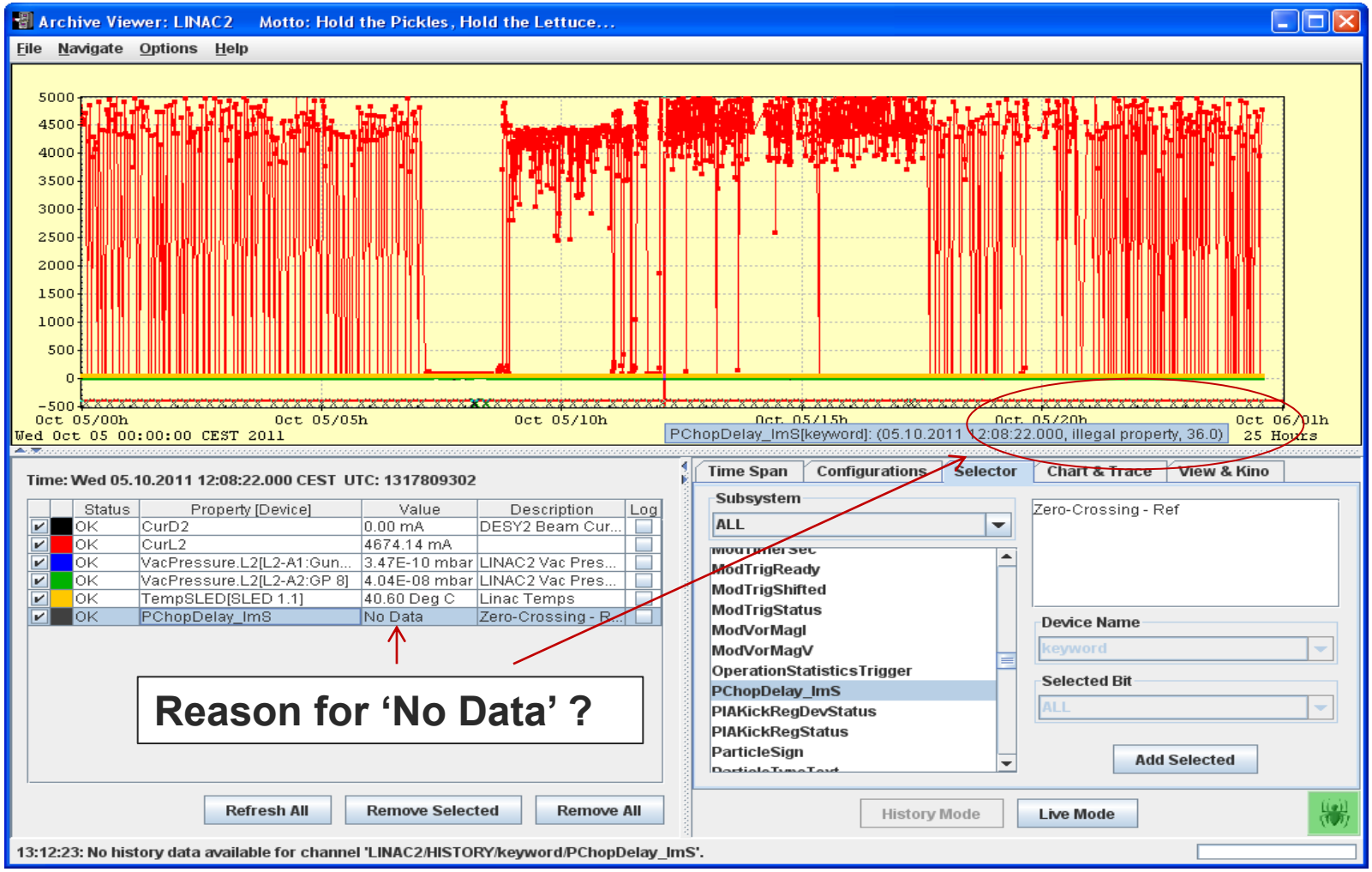
- Archive System
  - Central Archive
    - now accepts annotations
    - now stores link status (if non-zero)

Archive System

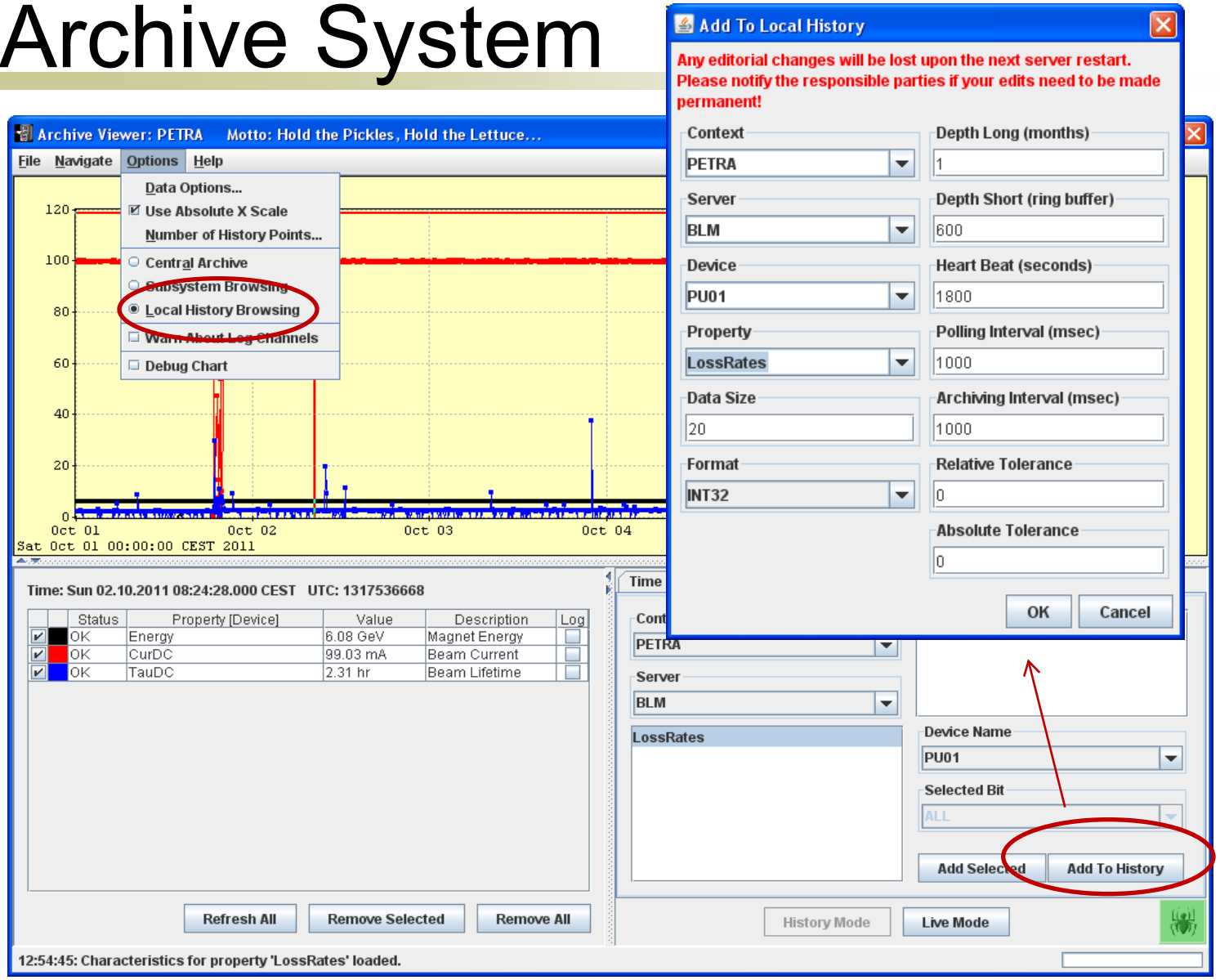
Found an annotation !



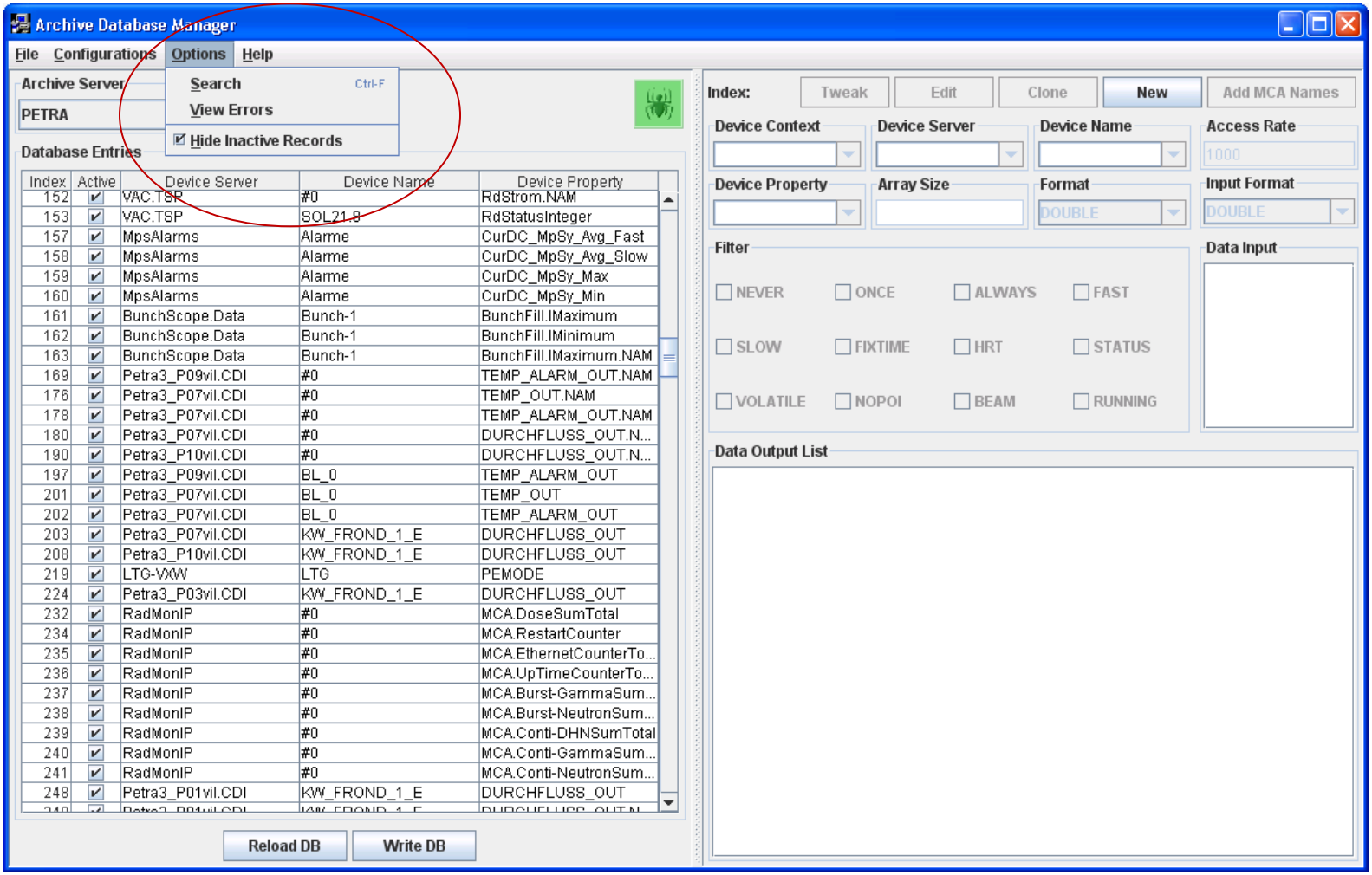
Archive System



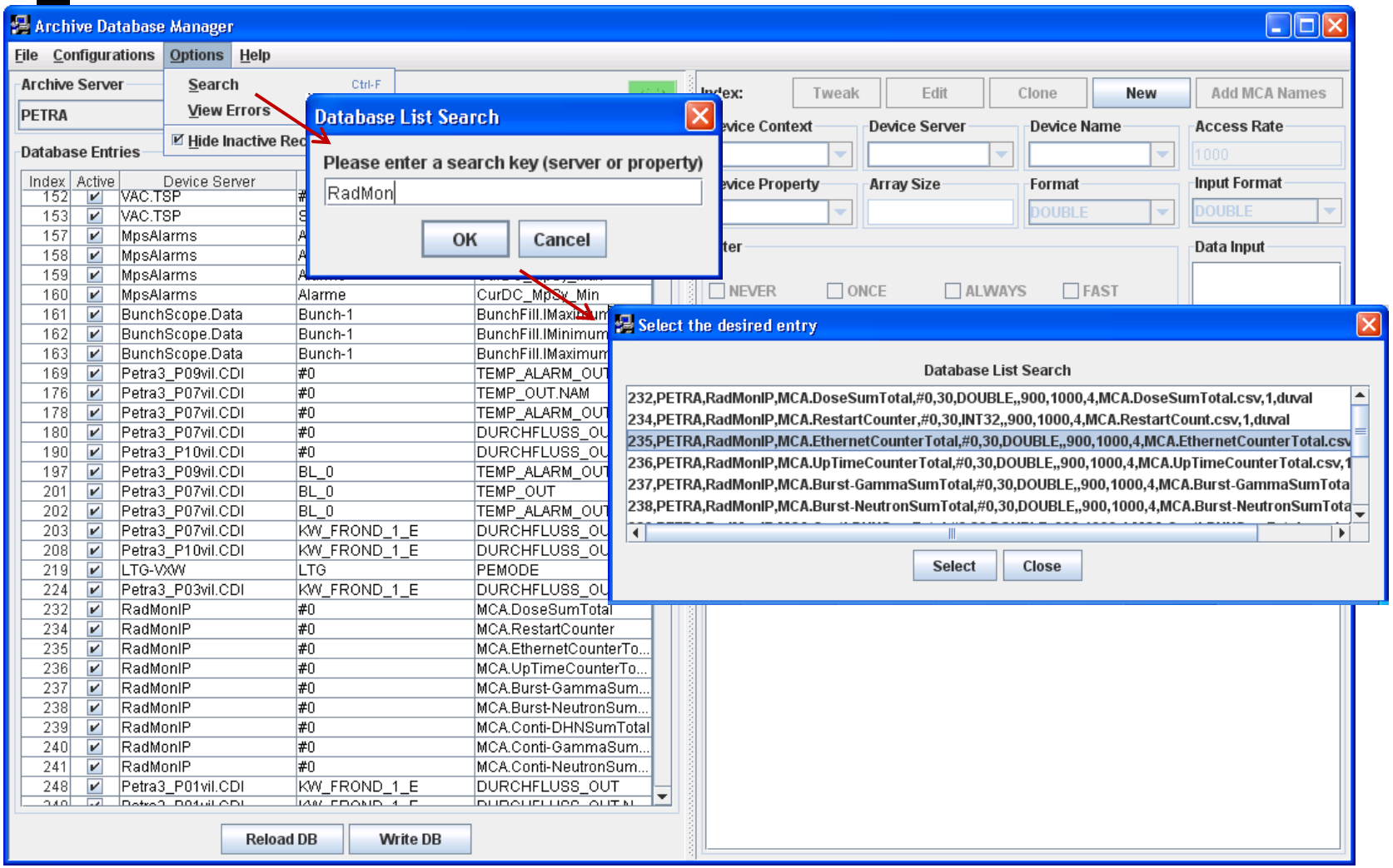
Archive System



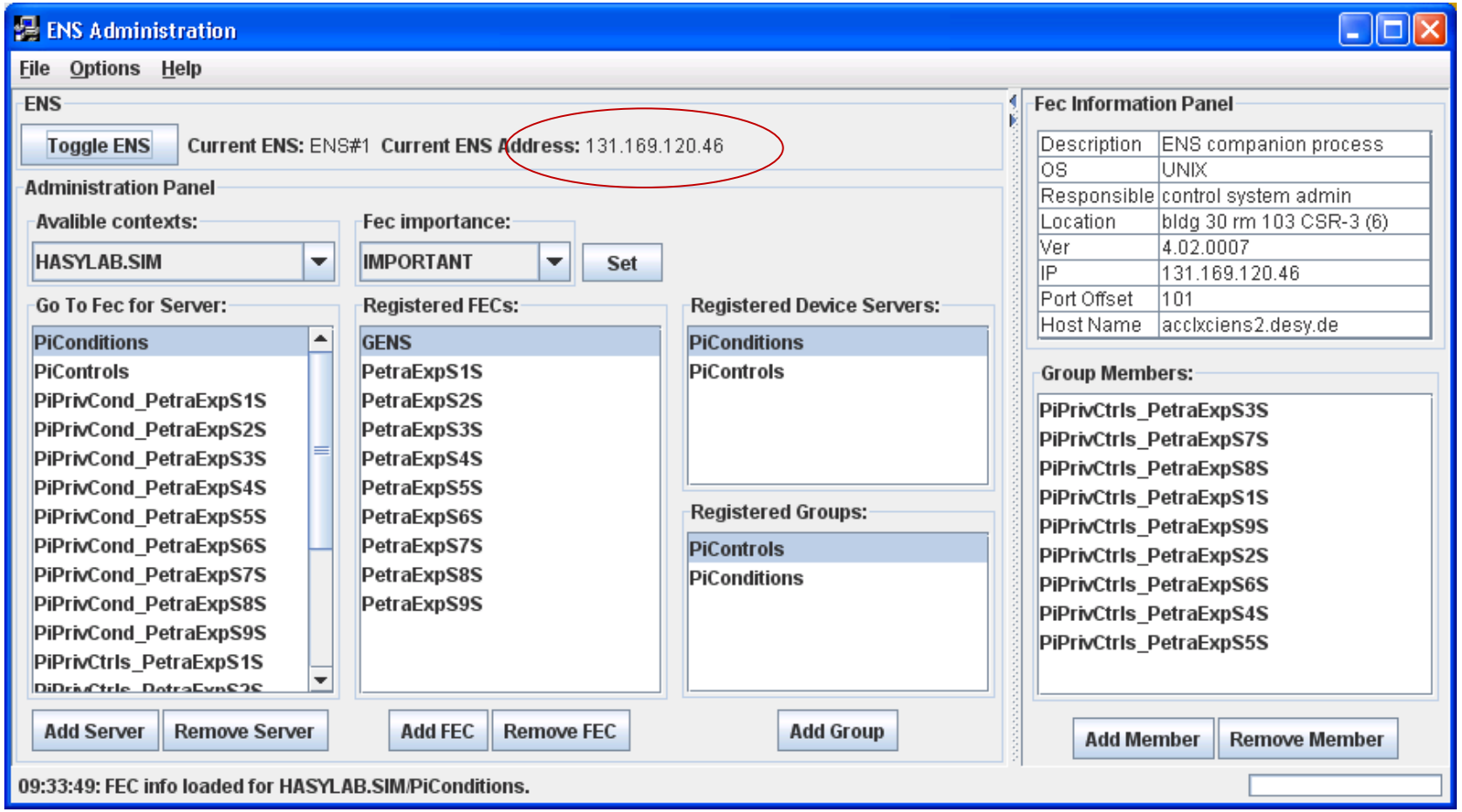
Archive Database Manager



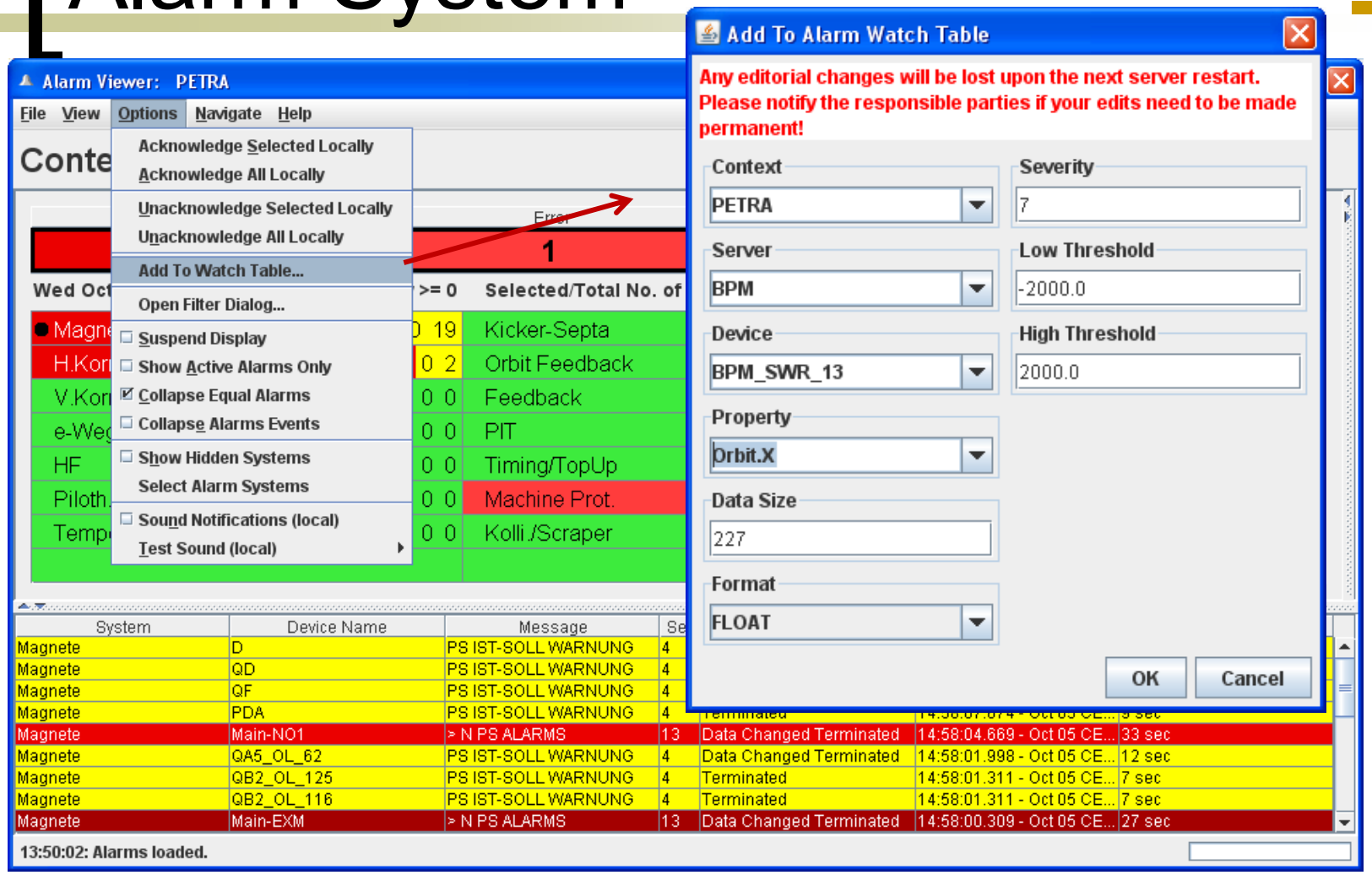
Archive Database Manager



ENS Administration



Alarm System



Alarm System