TINE Release 4.x.x News

(Feb. 8, 2017: That was the month that was !)

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

Release Notes 4.6.0

- (RE)-CONNECTIVITY IMPROVEMENTS: Numerous improvements in handling TCP connections problems have been introduced. In addition improvements to contract reconnection with transport modes such as CM_EVENT have been introduced.

Affects: client side links, mostly those using TCP mode and/or CM_DATACHANGE or CM_EVENT transport flags.

Possible side effects: None expected.
Attention level: GREEN

- CLEANUP IMPROVEMENTS: Overall resource and memory cleanup is now significantly improved when a dynamical (shared) tine library is explicitly unloaded from memory.

In the past, various lists and resources where left resident when a client-side application exited, with the expectations that all allocated resources are returned to the system. This is true if the application is the entity unloading the library when it exits. However, an application such as matlab or python will load the tine shared library when necessary and will unload the tine shared library if e.g. a 'clear mex' or 'del PyTine' is called. This action does not exit the application but does unload the library and leaves memory and resources unaccounted for (a memory leak) until e.g. matlab or python are themselves exited.
As of release 4.6.0 this is no longer the case: a 'clear mex' or 'clear tine' will unload the tine library and return all memory an resources to the system.

Note: There was no memory leak involving the tine library under normal operational conditions.

Affects: MatLab, Python and other applications explicitly freeing the tine resources.
Possible side effects: None expected.
Attention level: GREEN

Release Notes 4.6.0

- New Feature: The API call SetConnectionTableCapacity() is now 'dynamic'. Meaning: it can be called at any time (not just at initialization) and will accordingly re-allocate the connection table memory if the capacity is increased.

Affects: The client-side connection table size.

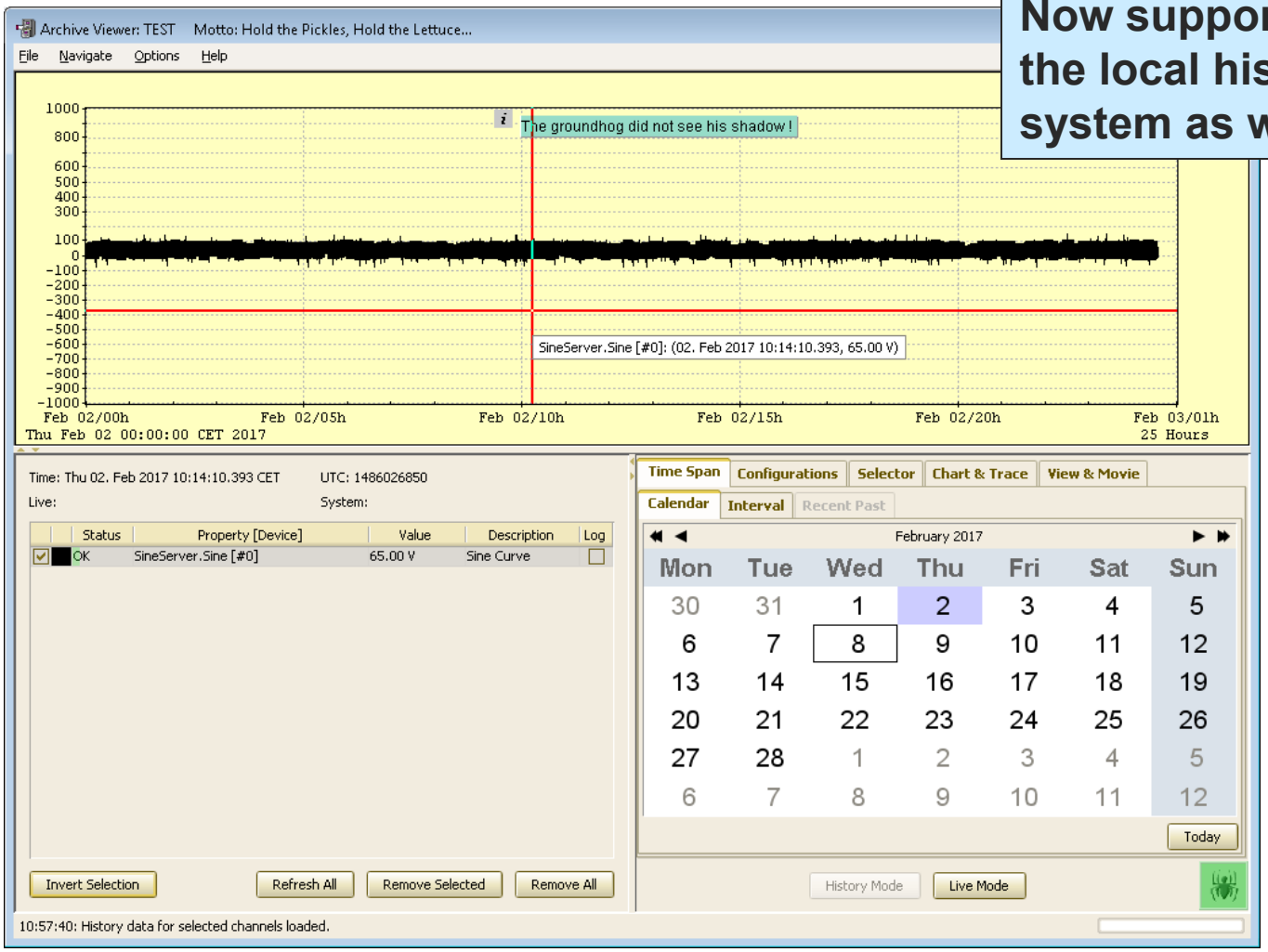
Possible side effects: None expected.

Attention level: GREEN

- New Feature: The local history subsystem now supports 'annotations'. Local history annotations refer to the entire device server and are not specific to any particular local history record. In conjunction with this new feature, the stock properties HISTORY.CMT and HISTORY.CMTS are also available.

Affects: Local history subsystem
Possible side effects: None expected.
Attention level: GREEN

Annotations



Now supported by the local history system as well …

Reconnection @ high rates

- **Normally:**
  - 1 Hz => Subscription package = 60 transfers
  - Subscription *counter* decremented for each transfer
  - Signal: *counter* = 10 -> signal for re-subscribing from client
  - @ higher rates => 'renewal multiplier' based on **N** Hz vs. **1** Hz.
  - Client does not renew -> server stops sending to him!
    - i.e. no 'dangling clients' allowed !
  - What happens if the server *schedules* the delivery at some external trigger rate which is systematically unknown?
    - schedule @ 10 Hz (or higher) but the contract requested a 1 Hz (or lower) polling interval ?
    - Not a lot of time for the client to react to the *signal* …

Reconnection @ high rates

int SetPropertySubscriptionRenewalLength (char * eqm,
char * prpName,
int value
)

Sets the current subscription renewal length for the property specified.

Persistent contracts established by a client calling one of the **AttachLink()** family 'subscribe' for the contract on the server [illegible] be renewed by the client. This of course happens automatically in the TINE kernel and is a guarantee that clients which 'disappear' ungracefully do not persist indefinitely. The 'number of responses' is known as the subscription renewal length. This value is adjusted according to a client's desired polling interval (i.e. the default value at 1 Hz, 60, is augmented to higher values when polling for instance at 10 Hz). The default value might under some circumstance be deemed to be too small. e.g. a server knows that properties are being 'scheduled' (see **_SystemScheduleProperty()**) at many Hz, which would cause the subscription counter to descrease more rapidly than the client's polling interval would otherwise dictate. In such cases the subscription 'renewals' would be far more frequently than an efficient data transmission would warrant.

Parameters:

eqm (input) is the local equipment module name (maximum 6 characters in length) For example: "BPMEQM".
prpName is the registered property for the buffer is to be assigned.
value is the desired setting

Returns:

0 upon success of a TINE error code

See also:

SetSystemSubscriptionRenewalLength()

References **feclog()**, and **GetPropertyListStruct()**.

Server side: set the renewal length for the scheduled property per API.

int SetSubscriptionRenewalThreshold (int linkId,
int thresholdInPercent
)

Gets the current client-side subscription threshold for the link in question.

Persistent contracts established by a Client API Callsalling one of the **AttachLink()** family 'subscribe' for the contract on th[illegible] and must be renewed by the client. This of course happens automatically in the TINE kernel and is a guarantee that clients[illegible] 'number of responses' is known as the subscription renewal length. This value is adjusted according to a client's desired po[illegible] higher values when polling for instance at 10 Hz). The default value might under some circumstance be deemed to be too s[illegible] 'scheduled' (see **_SystemScheduleProperty()**) at many Hz, which would cause the subscription counter to descrease mo[illegible] dictate. In such cases the subscription 'renewals' would be far more frequently than an efficient data transmission would w[illegible] threshold value by making use of this call.

Parameters:

linkId is the link id for the link in question (returned from the original call to **AttachLink()**.
thresholdInPercent is the desired threshold given as a percent of the total number of subscription transfers. This should be positive integer. Accepted values range between 10 and 90 (percent).

Returns:

0 upon success of a TINE error code

See also:

GetSubscriptionRenewalThreshold()

References **feclog()**.

Client side: set the renewal threshold for the associated data link (as a percent of total delivery).

Release 4.6.0

Normal Client-Server Communication vs. Exotica …

- *Normal*
  - e.g. server offers property "**P**" which is bound to a float variable
    - or some array of a *normal* data type.
    - property is registered with all relevant information
  - client attaches a link to property "**P**"
    - *asynchronous* (or *synchronous*)
    - specifies the preferred data type and size
    - specifies a polling interval
    - specifies a transfer mode (TIMER, DATACHANGE, etc.)
  - or sets property "**P**"
    - *synchronous* (or *asynchronous*)
    - Specifies a timeout
    - base transfer mode is SINGLE

Release 4.6.0

Normal Client-Server Communication vs. Exotica …

- The fun begins when …
  - **P** uses a *complex* datatype
    - e.g.
      - CF_STRING, CF_KEYVALUE
      - CF_SPECTRUM (CF_ASPECTRUM)
      - CF_IMAGE (CF_AIMGAGE)
      - CF_MDA
      - CF_STRUCT
      - CF_DBLTIME
    - More difficult to archive, save-and-restore, etc.
  - The caller uses CF_DEFAULT
    - signal for the server to send the (last overloaded) data type and size.

Release 4.6.0

Normal Client-Server Communication vs. Exotica …

- The fun begins when …
  - The server schedules **P** at some external (systematically unknown) rate.
  - The server *redirects* **P** or the device which supports **P** to some other server.
  - **P** is a multi-channel array
    - and some of the devices are redirected

Release 4.6.0

Normal Client-Server Communication vs. Exotica …

- The fun begins when …
  - The server is a member of a server *Group*
    - e.g. “/XFEL/LLRF.CONTROLLER”
      - composed of
        - “/XFEL/LLRF.CONTROLLER.1”
        - “/XFEL/LLRF.CONTROLLER.2”
        - “/XFEL/LLRF.CONTROLLER.3”
        - …
  - The client uses wild cards
    - e.g. calls “/XFEL/VAC.ION_PUMP/*[P]”
    - and the server is a ‘GROUP’ server.

Release 4.6.0

Normal Client-Server Communication vs. Exotica …

- The fun really begins when …
  - Many exotica happen at the same time …
    - The multi-channel property **P** is redirected for many of its registered devices.
    - The client makes a *wildcard* call and uses data type CF_DEFAULT.
    - (and imagine if some of the redirected-to servers are *scheduling* the property and others not! -> don't try this)

Release 4.6.0

- Redirections and the XFEL **MML**
  - The XFEL **CMS** and **MML** logic are on two separate servers
  - The CMS redirects almost all (all?) devices.
    - And there are lots of them ! (>700 at the moment)
    - 700 x ~7 properties per device => 4900 redirection entries
  - **MML** needs to learn *ALL* of this.
  - Initial problems:
    - **CMS** uses ‘deep’ redirection
      - The EQM handler knowns and provides the redirection information and NOT the device registration itself.
      - **MML** was issues set commands in a *bundle*.
    - The C-Lib was using a simple linked list as the redirection table
      - Java uses a hash table.

Release 4.6.0

- Redirections and the XFEL **MML**
  - C Lib now handle ‘deep’ redirections with a bundled call
  - C Lib now uses a hash table for redirections
  - But (best practice) …
    - MML now acquires and makes use of the redirection information at initialization
      - After all: it should have a priori knowledge of this anyway.

Release 4.6.0

- Python news:
  - PyTine now accepts a *tuple* as input data in a set or call !
    - e.g. when the input is FLTINTINTINT object.

Release 4.6.0

- Acop.NET status