TINE Release 5.0: A First Look

First major release in 10 years!
(but … there's no reason to go Fibonacci …)

P. Duval, J. Szczesny, T. Tempel, DESY, Hamburg, Germany; S. Weisse, DESY, Zeuthen, Germany
M. Nikolova, EMBL-Hamburg, Germany; J. Bobnar, Cosylab, Ljubljana, Slovenia

DESY

Abstract

The TINE control system evolved in great part to meet the needs of controlling a large accelerator the size of HERA, where not only the size of the machine and efficient online data display and analysis were determining criteria, but also the seamless integration of many different platforms and programming languages. Although there has been continuous development and improvement during the operation of PETRA, it has now been 10 years since the last major release (version 4). Introducing a new major release necessarily implies a restructuring of the protocol headers and a tacit guarantee that it be compatible with its predecessors, as any logical deployment and upgrade strategy will entail operating in a mixed environment. We report here on the newest features of TINE Release 5.0 and on first experiences in its initial deployment.

Upgrade Strategy

How to update a running machine …

- Release 5.xx must interoperate with Release 4.xx seamlessly!
- Make use of TINE Unit Server and Unit Client in multiple combinations
  - Check R5 <-> R4 and R5 <-> R5
- No amount of unit testing will catch everything!
- Multi-cultural machine control at DESY
  - 64 and 32-bit front end servers on both Windows, Linux, VxWorks.
  - Java, LabView, and Python servers …
- TINE is feature rich
  - Large variety of ways to do things
  - Entropy is large

- Test central services in mock environment …
- Test semi-important servers during operations …
- Adiabatically release during machine studies period following mini-shutdown
  - Servers and Clients making use tine.jar (Java) or shared libraries (C/C++) will automatically upgrade.
- Be prepared to rollback!
- Be prepared for extreme programming!

- The mere act of shaking things up will expose existing problems!
- Several hiccups repaired during machine studies
- Two others during the ensuing user-run
- TINE Release 5 is now in place and the de-facto standard at the PETRA III complex (still ~40% Release 4 servers in place).

TINE Features

TINE Release 4 and 5 both support the following features …

- Multi-Protocol
  - UDP, TCP, Shared Memory
  - (IPX is optionally available)
- Multi-Platform (clients and servers)
  - Windows, MAC, Linux (C/C++ or Java)
  - VxWorks, Solaris, other Unix Systems
  - DOS, VMS, other legacy systems
- Multi-Architecture
  - Transactions (client-server)
  - Publish Subscribe
    - Event Scheduling
  - Producer Consumer
    - Publish-Consumer
- Multi-Language
  - C/C++, Java, .NET, LabView, Python, Matlab
- Multi-Threaded
  - (Single-threaded builds also possible)
- Multi-cast
  - Producer Data is multi-casted
  - (can also be broadcasted)
- Address Redirection
- Compound Data Types
  - Various tuples in addition to the primitives
- Structured Data
  - User-defined structures for atomic transfers
- Contract Coercion
  - Servers can coerce client requests into efficient transfers
- Save-and-Restore
  - Initial property values use the last stored setting
- Local Histories
  - of designated properties
- Local Alarms
  - Specific or Automatic alarms
- Local Statistics
- Extensive Central Services
- Extensive Diagnostics

TINE Services

- Naming Services
  - Plug-and-play
  - Equipment Name Server is a TINE server (not LDAP!)
- Globals
  - Important Machine Parameters
  - Time Synchronization
  - Cycle/Event Numbers
- Logging
  - Central logging
  - Local Logging
- Alarms
  - Central Alarm Server
  - Fatal Alarm States
- Archive
  - Central Archive for Machine Parameters
  - Event Archive for post-mortem or other designated events
- States
  - Machine State counting
  - Operation History with state corrections and fault blaming
- Statistics
  - Front End Statistics
- Debugging
  - Attach to running servers or clients

Release 4 Issues

Needed to address the following issues in the Release 4 protocol …

- Protocol
  - Provide full support for IPv6
- Headers
  - Update message-size-in-bytes to unsigned 32-bit integer (was 16-bit)
    - aids in reassembly for jumbo transfers
  - Include process ID (client-side request header)
  - Include client type (client-side request header)
  - Include message-size-in-elements (server-side response header)
    - aids in generic call response interpretation
  - Include contract data size (server-side response header)
    - aids in generic call response interpretation
  - Include endianness and character encoding
- API
  - No major API issues
  - Some refactoring needed in C library to avoid name collisions in STL or MFC.

Release 5 Solutions

- Protocol
  - IPv6 fully supported (as of 4.6.3)
    - Use dual stack when possible.
    - Fewer headaches with an evaporating address space
    - Real jumbo-grams (4,294,967,295 bytes) possible
    - BUT: IPv4 still very common …

    > And who can remember an address like fe80::7c89:716f:a87f:38da anyway ?
- Headers
  - message-size-in-bytes now unsigned 32-bit integer
  - process ID available (client-side request header)
  - client type available (client-side request header)
  - message-size-in-elements available (server-side response header)
  - contract data size available (server-side response header)
  - Include endianness and character encoding
    - still fixed as LITTLE ENDIAN and ASCII
  - Data Stamps still 4-byte unsigned integers
    - timestamp is an 8-byte double
    - System stamp (event/cycle number) is 4-byte integer
      - (will wrap in 14 years at 10 Hz)
- API
  - Many macro #defines are now enums.
  - C++ and STL or MFC:
    - can optionally make use of namespace wrapper around tine.h or not :

    ```
    namespace tine
    {
    #include "tine.h"
    }
    ```
  - API unchanged as to the library calls themselves.

A sampling of a server report …

More extensive diagnostic information …

client side

server side

---

it doesn't always have to have a *slick* web site to be good …