TINE RELEASE 4 IN OPERATION

Philip Duval, Piotr Karol Bartkiewicz, Steve Williamson Herb, Honggong Wu, DESY, Hamburg
Stefan Weisse, DESY, Zeuthen

Abstract

The TINE [1] 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 was a determining criterion, but also the seamless integration of different platforms and programming languages of the many applications developers. In keeping pace with new technologies and the new generation of accelerators such as PETRA3, FLASH, PITZ and associated pre-accelerators and beam lines, TINE has undergone a major "face-lift" in its most recent version, 4.0.4, where platforms such as Java and LabView are not only supported, but emphasized. In addition, TINE Release 4 integrates video transport, the device layer, and central services to a much greater extent than its predecessor. We report here on many of the new features and how they are currently being used in operations.

INTRODUCTION

Originally a spin-off of the ISOLDE control system [2], TINE is now a mature control system, where a great deal of developmental effort has gone into the control system protocol itself, offering a multi-faceted and flexible API with many alternatives for solving data flow problems. As the TINE kernel is written in straight C and based on Berkeley sockets, it has been ported to most available operating systems. By the same token, a native java port of TINE is currently being used extensively in the new control systems of PETRA3 and its pre-accelerator chain, as is LabView.

Over the past couple of years much effort has also been spent in providing standardized access to hardware. This has largely been realized through use of the Common Device Interface (CDI) [3], the TINE Can Open Manager (TICOM) [1,4], and the TINE Network Queue [1].

In order to keep pace with developers' demands, correct noted deficiencies, and take maximum advantage of the technological advances in networks and communications, TINE Release 3.31 has been 'put on ice' and has given way to Release 4 (currently 4.0.4). We shall describe below the additions, enhancements and improvements that are now to be found in TINE Release 4.0.4.

OLD FEATURES

Before reporting on 'what's new', we review some of the 'old' features which, if not unique to TINE, are certainly not mainstream in the controls community.

Multicasting

TINE offers at the API level a variety of data transport mechanisms. The developer of course does not need to know about any of these, as the default chosen by the TINE kernel is in most cases appropriate. However under certain (extreme) scenarios, making use of an alternative transport can have a tremendous impact on scalability and efficiency. For instance, when multiple clients are eating up the network bandwidth or CPU load at the server, a good solution is to use TINE multicasting, where all clients essentially collapse to a single 'network' client. Another use of multicasting corresponds to a producer-consumer architecture, where the server simply sends certain data into the system via multicast. This might include beam and state parameters of the facility and possibly cycle or pulse numbers.

Data Types

TINE servers can send data with any of the standard primitive data types. A large set of 'complex' data types is also available, whereby doublets, triplets or even quadruplets of data can be sent atomically (e.g. a value-status pair). In addition to this, the server developer can define, register and use any structure type he desires using the 'tagged structure' feature.

Multi-Channel Arrays

In addition to data types, a data object can have an 'array' type. This is in many cases 'scalar' (single value) or 'spectrum' where the data object delivers for instance a scope trace. TINE also allows a kind of serialization where a data object can have array type 'channel'. In such cases the data consists of an array containing the property values of all devices (of necessity having the same settings and units). This is an enormously practical feature for such entities as vacuum pressures, beam position monitors (orbits), magnet PSC currents, etc. as these can then be archived and retrieved as such.

Stock Properties

All TINE device servers have a standard set of properties called "stock" properties which not only allow information to be queried but in some cases manage access control (such as "ACCESSLOCK"). In addition, a standard set of property-specific meta-properties are also available. Thus the history of a given property, or a list of the devices implementing a specific property, can be obtained.

NEW FEATURES

Data Objects

The major change in TINE release 4.0 concerns the supported name lengths, where registered device and property names for instance can now contain up to 64 characters.

TINE data objects now also contain, in addition to the data timestamp, a system supplied data stamp, and a server supplied data stamp. Data returned to a caller also carries the transfer 'reason', and so can be queried as to why the data was sent (data change, heartbeat, event, etc.).

Name Space Changes

TINE is now systematically case insensitive regarding name resolution. In addition, although the maximum length for a registered device name is 64 characters, the release-4.0 protocol supports sending device names with extended lengths up to 1024 characters. Device 'names' of such lengths are of course not registered but are sometimes used for instance when supplying a device 'list' to a CDI server or when accessing the stock property "SRVLOGFILE" where the device name is used to carry the file name and path specification.

Redirection

In TINE Release 4.0, a device server can redirect a call to a completely different hierarchy of context, server, device, and property. This is a significant improvement over release 3.31 where the context (top of the hierarchy) and device name were not allowed to change.

Tagged Structures

User defined tagged structures can, as of Release 4.0, embed any other TINE format type as well as other tagged structures.

Bitfields

Individual bits and bit fields contained within Integer data can now be systematically encoded and acquired using TINE bitfields. Registered bitfields are 'tagged' and are registered in a similar manner as tagged structures.

Images and Video

Sending video via TINE is now more tightly integrated into the system. A new data type CF_IMAGE systematically specifies a video frame and all associated characteristics. Other features of release 4.0 have been honed to provide the most efficient video transfer possible. This includes API calls to massage the maximal transport unit (MTU) of UDP datagrams as well as calls to allow 'deep' data binding of a video server's image frames (i.e. no double buffering).

Multicasting

Multicasting in release 4.0 no longer uses a single multicast group for all servers but now makes use of server-specific multicast groups. In this case the last 2 bytes of a server's IP address identify the server's multicast group. This is used both in publish-subscribe style multicasting as well was in sending TINE globals via producer-consumer style multicasting

Alarms

The alarm messages in TINE release 4.0 now contain a timestamp with millisecond precision and (more importantly) the start time of the alarm. A TINE alarm belongs to a device and has a 'history'. An active alarm has a start time and is periodically re-issued according to circumstances (heartbeat, alarm data change, oscillation) until the alarm is terminated. If the same device then re-issues the same alarm, it is regarded as a new alarm and will have a new start time.

Alarms can now carry up to 64 bytes of alarm-specific data, a much-needed improvement over the 6 bytes available to release 3.31 alarms.

Local Histories

The local history subsystem has also been much improved in that it is now possible for a device server to save worst-case, non-fragmented local history files. Here, 'worst-case' signifies a large file with room to handle all records at the given access rate with no filtering. This allows rapid archive data retrieval over long time periods even on NTFS, where file fragmentation can seriously impair performance. In addition, when operated in a multi-threaded environment, calls to selected stock properties (local history call among them) are handled on a separate thread, independent of normal server activity.

Server Configuration

Device servers can be configured by API or more commonly by configuration file(s). When configuration files are used, this has traditionally been a set of comma separated values (CSV) files. This is still possible for release-4.0 servers, but can now be more logically organized on the local file system. Furthermore, in place of the CSV files, a single XML configuration file can supply all or part of the server's configuration parameters.

CENTRAL SERVICES

The TINE central services have all been upgraded to release-4.0 standards and in most cases have been modified to take advantage of some of the new features.

Alarms

The Central Alarm Server (CAS) in a given context monitors the alarms of a relevant set of device servers (a list maintained by a control system administrator). It is responsible for the final stage of alarm filtering, alarm archiving and establishing 'server-down' alarms. Now that the TINE alarm message always contain the alarm stop time, the logic necessary to determine an alarm's duration is trivial.

Archive

Besides the local history archiving which can occur directly at the device server, TINE offers a central archiving system and an event based archive system. The former concerns itself with the regular archiving of machine parameters and latter with occasional archiving of, for instance, post-mortem data following an event signal.

Globals

Certain aspects of a TINE control system are usually provided via multicast in a pure producer-consumer mode. A given TINE context has a designated globals server which sends the (pre-configured) relevant beam parameters out at a predefined rate (typically 1 Hz). Similarly, TINE time synchronization requires a time server multicasting the designated system time at 1 Hz.

Recently, another entity specific to a given context can also exist, namely the "cycler.", which multicasts a system cycle number, to be used as a systematically data stamp in a similar vein as the time server.

CONFIGURATION AND RAD TOOLS

A number of tools are available to aid the development of both server and client applications.

At the device server level, a significant number of the servers now in operation at LINAC2, PIA, and DESY2 use CDI for interfacing to hardware. CDI servers are implemented by constructing a hardware data base; no custom server code need be written. Many of the remaining device servers are written in java and make use of a TINE java device server code generator [5]. Another category of device server is written entirely in LabView and makes use of the device server wizard to generate the appropriate TINE configuration database. In such cases the developer can concentrate on the business logic and not worry about interfacing to control system clients.

At the client level, most new control system applications are being written in java. A tremendous help in writing these applications has been the advent of ACOP beans[6], developed in collaboration with Cosylab [7] which allow design-time browsing when a visual editor is available and which generate relevant code when connecting to an endpoint. Recently, the set of ACOP beans has been expanded to include a video bean, which integrates directly into the TINE video system.

COMMISSIONING

Beginning in July 2008, the LINAC2 accelerator used for injecting into DESY2 and ultimately into DORIS and PETRA3 came on line with many new control system components, both hardware and software. Many of the hardware components using the in-house SEDAC bus were replaced with new equipment using the CAN bus. The Common Device Interface (CDI) was likewise getting its first taste of operations. And, the control system moved from TINE release 3.31 to TINE release 4.0. In addition, most applications moved from windows to java on the client side, and a significant number moved from windows to java on the server side as well.

It is not surprising that bugs, unforeseen use-cases, incomplete implementations, lack of familiarity, and insufficient diagnostic tools all led to occasional controls problems. Fortunately, there was already some operational experience with TINE 4.0 in the FLASH accelerator; there is a high degree of interoperability between TINE and the DOOCS [8] control system used for FLASH, and some FLASH components, such as magnet power supplies, use TINE device servers.

Over the initial few weeks of operations, many bugs were found and eliminated, the most insidious involving concurrency problems with multiple threads. Equally challenging was the need to sift through large amounts of data quickly and consistently, as was the case with the alarm viewer client during commissioning when it was being handed thousands of alarms by the central alarm server.

Nonetheless, as of mid October 2008, all known controls problems have been tracked down and eliminated, and LINAC2, PIA and DESY2 are running stably with the new control system hardware and software.

The interconnectivity to other control systems relevant to PETRA III is either in place or ready for commissioning. As alluded to above, DOOCS and TINE can operate seamlessly together, so there is no hesitation to re-use for PETRA, DOOCS servers used for some of the vacuum elements in FLASH. Many power and infrastructure channels running EPICS are likewise easily integrated via Epics2Tine[8]. Most recently, Tango2Tine[9] has been upgraded to support TINE Release 4. This will be a useful component when exchanging data with certain elements in the HASYLAB beam line control.

REFERENCES

[1] http://tine.desy.de

[2] "A PC Based Control System for the CERN ISOLDE Separators", R. Billings et al, ICALEPCS '91.

[3] "Using the Common Device Interface in TINE", P.Duval and H.Wu, PCaPAC'06.

[4] "Integration of CANopen-based Controllers with TINE Control System for PETRA III", P.Bartkiewicz, et al, ICALEPCS 2007.

[5] "First Experiences with a Device Server Generator for Server Applications for PETRA 3", J. Wilgen, these proceedings.

[6] "The ACOP Family of Beans: A Framework Independent Approach," J.Bobnar, et al, ICALEPCS 2007.

[7] http://www.cosylab.com

[8] http://doocs.desy.de

[9] "An EPICS to TINE Translator", Z.Kakucs et al, ICALEPCS 2001.

[10] "EPICS to TANGO Translator", R.Stefanic and L.Geoffroy, ICALEPCS 2007.