DOI 10.5162/ettc2020/2.5

Addressing the Babel's Tower of FTI Standards in a Network Environment

Patrick Quinn

Aerospace Instrumentation
Curtiss-Wright
Unit 5, Richview Office Park, Clonskeagh, Dublin 14, Ireland
pquinn@curtisswright.com

Abstract: This paper discusses the different demands the many industry wide standards place on flight test instrumentation hardware and software in a networked environment and the challenges of supporting all standards from a supplier point of view.

Keywords: Paper format, instruction to authors

1. Introduction

Today's networked flight test instrumentation (FTI) hardware, and supporting software, needs to be a Babel Fish, the universal translation device from Hitchhiker's Guide to the Galaxy, to support the multitude of industry standards.

From Chapter 10, TmNS, iNET-X, IENA and DARv3 transmission protocols to TMATs, MDL, XidML and XML metadata, both the hardware and software are required to speak and understand multiple packet types and file formats.

Each of these formats present their own challenges and have their own advantages and shortcomings, and each present different challenges in a distributed networked architecture.

In an ideal world, connecting networked FTI systems would be a simple – just "plug and play". However, experienced users will tell you that this is just not the case.

This paper discusses some of the challenges imposed on FTI, both at the hardware and software levels, by these various standards and highlights how these may be addressed.

2. Chapter 10 & TMATS

IRIG-106-Ch10 is probably the oldest standard of all the ones discussed in this paper, and probably the one that imposes the largest demands on a modern networked FTI system.

The standard itself was originally a solid state recording standard that evolved from the move away from tape based recorders in the late 1990's. It has evolved over the years to add more data types and time formats.

The Chapter 10 standard is inherently a recording standard. It is not network centric, it does not define how any of the Ethernet protocols, like TCP/UDP/QOS etc., should be leveraged to optimize the FTI network.

It defines a specific data structure which the data must be recorded. The structure of the Chapter 10 recording file is presented in Figure 1.

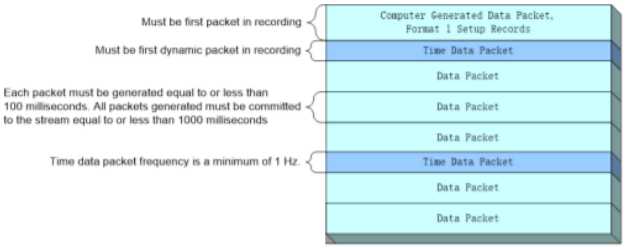

Figure 1: The Chapter 10 format

The first packet in the recording is the Computer Generated Data packet that describes all the subsequent data packets in the recording. This is usually the TMAT's file.

2.1 TMATS

In a networked distributed architecture there are two approaches to using chapter 10 as your data standard:

1. Record all data acquisition units (DAU) to a single system wide chapter 10 recording
2. Record all DAUs to individual chapter 10 files per DAU

For the TMATS file the challenge here, when using approach 1, is primarily a software one. The FTI configuration software must be capable of describing all the data captured by all DAUs into a single coherent TMATS file that describes all packets in the system.

When using approach 2, this is simpler, but the configuration software must generate a TMATS file per DAU and keep them all up to date as the configuration changes.

2.2 Time Packets:

The real challenge at the hardware level when using Chapter 10 in a distributed networked architecture is the time packet.

The first dynamic packet in the recording must be a time packet, this time packet must per periodic and repeat at a minimum of 1 Hz.

Chapter 10 defines 3 formats of time packet, Format 0 is reserved for future use, Format 1 covers time from GPS / relative time counter and Format 2 covers network time.

Format 1 contains a channel specific data word (CSDW) that indicates the IRIG time source, covering everything from free running to PTP locked to external embedded time sources. The resolution of the time stamp in Format 1 is down to milliseconds.

Format 2 also contains a CSDW that indicates the network time format (NTP, PTPV1 or PTPV2) and the validity of the network time. It has a time resolution down to nanoseconds. Format 2 would seem to be the ideal time format to use in networked systems.

However, there is one major missing piece of the puzzle that needs to be addressed for chapter 10 to be truly useable in a distributed networked architecture, this is the question of "Who creates the time packet?" and "How does this time packet keep track the time from all the other DAUs?".

With multiple DAUs, each DAU can produce its own time packet. When these time packets arrive to the recorder the recorder must decide which time packet to use to time stamp the full system recording. There are multiple possible solutions to this:

1. The system could be configured such that only one DAU actually produces a time packet and the recorder uses this time packet for its time source.
   - This has the advantage of being a simple approach, but it does not reflect the status of the data from the other DAUs relative to the time master DAU
   - What happens if Time Master DAU goes out of PTP lock?
   - What happens if Time Master DAU time packet is dropped or delayed?
2. The recorder could produce its own time packet
   - Again a simple approach, but with the same issues
3. The recorder could track the time packets from all DAUs and make a decision on which is the best.
   - Difficult to implement
4. The recorder could record each DAU to its own individual chapter 10 recording
   - Not an efficient use of recording bandwidth
5. The time packet requirement could be dropped by the standard
   - The secondary header time could be used in post flight to align the data

2.3 Chapter 10 Data Types

Another feature that chapter 10 places on FTI hardware, and software, are the excess of different data types and different sub formats of each data type that must be supported for a true Chapter 10 compliant FTI set up. As of the IRIG-106-2019 release, the data types and versions of each are supported by the chapter 10 standard as noted in Table 1.

Data Type	Formats Defined
Computer Generated Data	Format 0, 1, 2, 3 & 4
PCM Data	Format 0, 1 & 2
Time Data	Format 0, 1 & 2
MIL-STD-1553	Format 0, 1 & 2
Analog Data	Format 0 & 1
Discrete Data	Format 0 & 1
Message Data	Format 0
ARINC-429 Data	Format 0
Video Data	Format 0, 1, 2, 3 & 4
Image Data	Format 0, 1 & 2
UART Data	Format 0
IEEE 1394	Format 0 & 1
Parallel Data	Format 0
Ethernet Data	Format 0 & 1
TSPI/CTS Data	Format 0, 1 & 2
CANBUS Data	Format 0
Fibre channel Data	Format 0 & 1

Table 1: **Chapter 10 data types**

Each of the above formats Chapter 10 also defines a UDP Transfer Header, which has 3 different formats,

- Format 1: Little Endian, 24 bit UDP Sequence number
- Format 2: Big Endian, 24 bit UDP sequence number
- Format 3: Little Endian, 16-32 bit Datagram Sequence number.

Each data type has their own sub set of rules around how the data is packed by the FTI hardware. For example, there are 6 different ways of recording PCM data just in PCM Data format 1 alone.

4. IENA

IENA is the Airbus network packet protocol that originated during the A380 program and has been widely adopted industry wide since then. Originally conceived as a network standard, the IENA protocol is quite simple and clear to understand, and support, from a networked architecture point of view.

Some of the restrictions IENA places on FTI hardware include

- Destination MAC must be Multicast, in the range 01:00:5E:01:01:00 - 01:00:5E:01:01:FF
- Packet fragmentation is allowed.
- Source IP must be in the format 172.28.X.X
- Destination IP must be in the format 235.1.1.X
- Source port must be greater than 50000
- Destination port must equal 51000.
- IENA time stamp is the number µS since the start of the current year.
- There are two status sections of the IENA header:

- Key Status – fixed for any IENA Key
- N2 Status – dynamic for any instance

- The sequence number is 16 bits
- There is an END word at the end of each packet, must be the same for all packets in any configuration

4.1 IENA Packet Types:

IENA defines 5 parameter types, and it is forbidden to use different parameter types in different IENA Keys.

4.1.1 P Type – Positional Parameters

Multiple occurrences of P Type parameters can be placed in one packet, but must follow a repeating pattern, as shown in Figure 1.

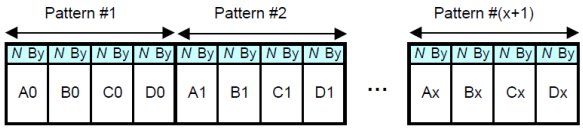

Figure 2: P Type parameters must follow a repeating pattern

One of the restrictions this places on any decom software is that it must know exactly how many parameters are in each packet and the occurrences of each in order to be able to locate all samples of any individual parameter.

4.1.2 D Type – Standard parameters with a delay field

These are groups of a maximum of seven 16 bit data words with an assigned parameter id and a 2 byte delay field. The parameters must be placed in a particular order and the delay field is the delay in µS from the packet time stamp to the acquisition time.

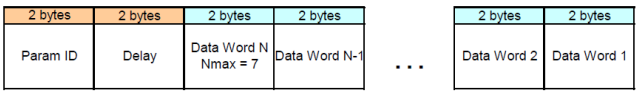

Figure 3: IENA D Type

4.1.3 N Type – Standard parameters without a delay field

Same as above, but without the elapsed time field.

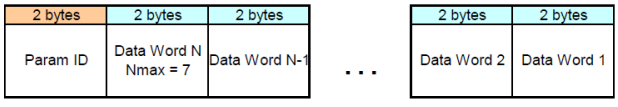

Figure 4: IENA N Type

4.1.4 M Type – Message parameters with a delay field

These are Message parameters whose length and acquisition time of the entire message can be reflected in the IENA packet. The data set can be padded if required and the delay field is the delay in µS from the Packet time stamp to the acquisition time.

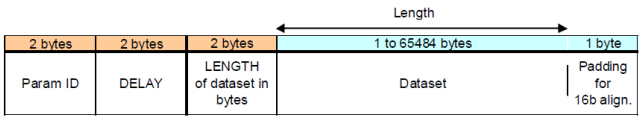

Figure 5: IENA M Type

4.1.5 Q Type – Message parameters without a delay field

Same as above, without the delay field.

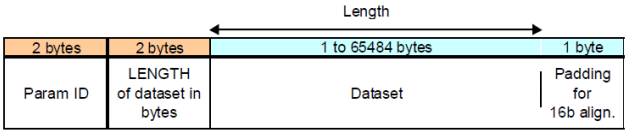

Figure 6: IENA Q Type

4.2 IENA shortcomings:

While IENA is popular there are a couple of shortcomings in the standard that do not take full advantage of todays networked systems capabilities:

1. Time stamp is in µS, in both the packet time stamp and the delay fields
   Greater time resolution is possible with other formats, such as IEEE 1588 PTP which allows time resolutions of nanoseconds to be reflected in the data
2. Parameter placement restrictions do not allow multiple samples of the same parameter to be placed contiguously

5. iNET-X

iNET-X is a network packet protocol developed by Curtiss-Wright's Dublin Office (formerly Acra Control). Again, originally conceived as a network standard, the iNET-X protocol evolved out of the early iNET (TmNS) definition and is also simple and clear to understand and support from a networked architecture point of view.

iNET-X does not support quality of service (QoS) protocols like pause frames, it does not allow packet fragmentation and does not support IGMP or DHCP. All packets are UDP packets, so TCP traffic is not supported. It does support aperiodic traffic transmission and SNMP. File transfer protocols like TFTP are supported, but not FTP.

From a network architecture point of view, iNET-X expects minimal dynamic behaviour from the sources and hardwired traffic routing. This requires the configuration software to be able to map the traffic flow throughout the entire FTI installation.

From a recording point of view iNET-X prefers open standards like PCAP and FAT32 file systems.

Unlike IENA, iNET-X allows:

- Both multicast and unicast destination MACs
- There are no restrictions on source IP, source port, destination IP and destination port

- The iNET-X time stamp takes full advantage of the nanosecond resolution offered by PTP-1588, with 64 bit time stamps counting from 1/1/1970
- There is a 32 bit flags field in the iNET-X header, which is used to dynamically reflect status of the data in the packets
- The sequence number is 32 bits

5.1 iNET-X Packet Types

iNET-X defines 4 packet types: placed, bit-aligned, block aligned, parser aligned and event.

5.1.1 iNET-X placed

This uses fixed, constant length packets, not exceeding 1426 bytes of payload. They must end on a 16 bit boundary. Multiple occurrences of any type of parameters can be placed in one packet, and multiple occurrences of parameters are placed contiguously.

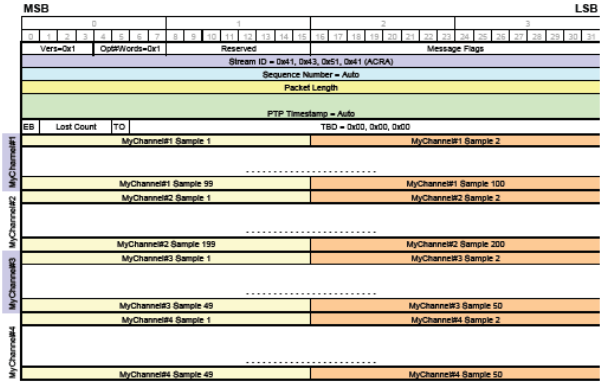

Figure 7: The iNET-X placed packet type

5.1.2 iNET-X bit-aligned

They have variable length packets, used for CVSD audio packets. Max payload length of Nx4 bytes, not exceeding 1426 bytes.

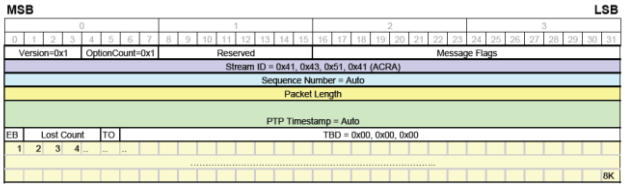

Figure 8: The iNET-X bit-aligned packet type

5.1.3 INET-X block-aligned

These are variable length packets, used for video transport streams. Usually constructed of X blocks per packet, with fixed number of bytes per block.

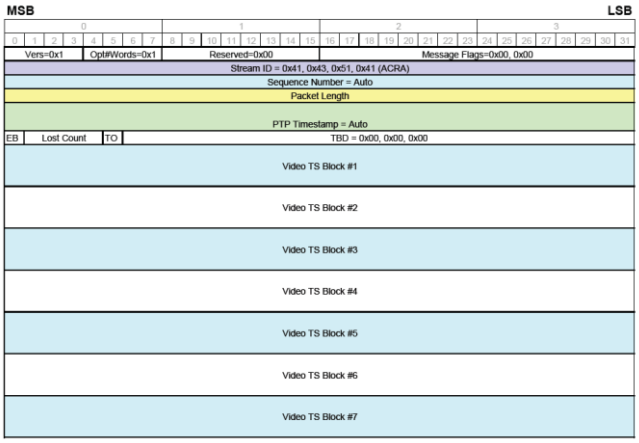

Figure 9: The iNET-X block-aligned packet type

5.1.4 iNET-X Parser-Aligned

These are variable length packets, used for bulk bus data capture. Each parser aligned block is a complete bus message, with elapsed time timestamp relative to the first message in the packet.

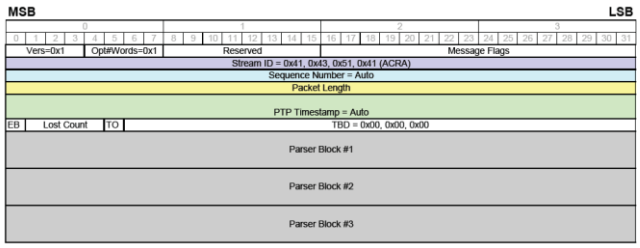

Figure 10: The iNET-X parser aligned packet type

5.1.5 iNET-X event

These are variable length packets, used for event marking. Usually made of a string representation of event code and a time stamp of the event.

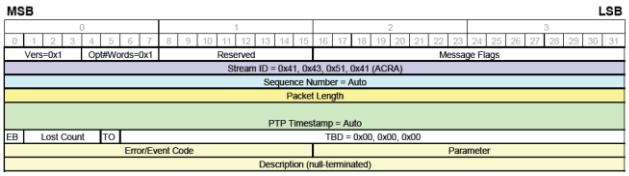

Figure 11: The iNET-X event packet type

6. DARv3

DARv3 evolved out of the DAR standard, originally developed by Boeing and the South-Western Research Institute (SWRI) for the Boeing 787. It was adopted by Curtiss-Wright's Newtown business unit (formerly TTC) and expanded upon to create DARv3. Once again, this protocol started life as a network protocol and is also simple to understand and support from a distributed networked environment.

DARv3 does support QoS protocols like pause frames, it does allow packet fragmentation, it does support IGMP but not DHCP. All packets by default are UDP packets, but TCP traffic is allowed. It does support periodic traffic

transmission and SNMP. File transfer protocols like TFTP and FTP are supported.

From a Network architecture point of view, DARv3 expects some dynamic behaviour from the sources, they must be able to respond to pause frames, and must advertise that fact to downstream devices during link negotiation. This requires the configuration software to be able configure a maximum latency for the packets to manage the traffic flow throughout the entire FTI installation.

DARv3 uses multicast packets and devices that support it use IGMP to join or leave multicast groups. Multicast packets use a default destination port of 50001, but this is configurable.

DARv3 supports fragmentation of jumbo frames, with fragmentation driven by message latency settings. DARv3 is big-endian. DARv3 also defines a network recording standard, the direct structure of which is compatible with the standard used for IRIG-106 Chapter 10, STANAG-4575 Edition 1.

The DARv3 packet format is made up of the DARv3 header followed by 1 or more data segments as shown in Figure 12.

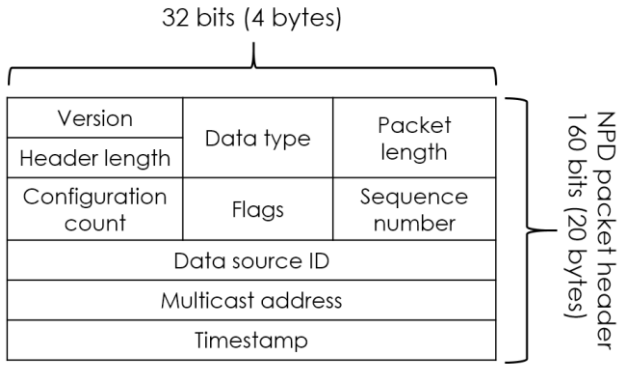

Figure 12: DARv3 packet

Unlike iNET-X, the time stamp, while based on IEEE-1588, is only the 32 bit seconds portion of the full 1588 time. The following data segments also follow a common structure and are all 32 bit aligned.

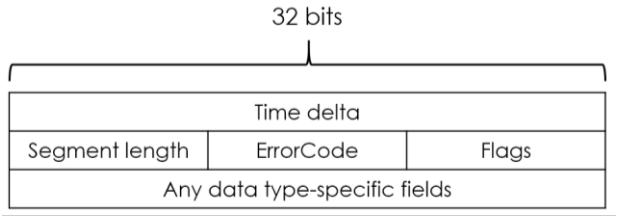

Figure 13: DARv3 time stamp

Here, the time delta is the nanosecond portion of the full 1588 time stamp, relative to the time stamp at the start of the packet.

6.1 DARv3 Packet Types

Similar to Chapter 10, DARv3 has multiple packet types, dependant on the type of data being captured. The data type being used is indicated in the header of the packet. It is therefore forbidden to mix data types in the same packet. As of the 2016, DARv3 standard the packet types in Table 2 are defined, each with its own set of rules around how that traffic type must be encapsulated.

Name	Protocol
1553	MIL-STD-1553
A429	ARINC 429
ACQ	Analog / digital signal acquisition
A664	ARINC-664
AUD	Audio
ETH	100/1000BT Ethernet
Event	Event packets
FAAD	Fwd Area Air Defence Standard
FC	Fibre Channel
HFCI	IMB protocol
HRV	High Res Jpeg-2000 Video
IP	IP Packets
Link-16	TDL 16
LRV	Low Res Jpeg-2000 Video
MP2	MPEG-2 Video
NDO	Arinc-664 Network Data Objects
PCM	PCM to IP gateway
Raw Bayer	High speed camera images
RDR	Radar
RGB	RGB video
Status	Status packets
TCP	TCP packets
UART	Serial Data
UDP	UDP Packets

Table 2: DARv3's packet types

7. TmNS

The Telemetry Network Standards Definition (TmNS) is the newest of all the standards and one of the most comprehensive in its attempt to define all aspects of the networked environment for FTI. It comprises of eight chapters of the IRIG 106 standard, covering everything from message formats to system configuration and management to RF networks.

TmNS does offer some new capabilities to the FTI environment that are really interesting:

1. Bi directional telemetry
2. PCM back fill
3. Dynamic spectrum sharing
4. Quality of service
5. Fully interconnected systems
6. Over the horizon telemetry

TmNS device support is truly comprehensive in that it covers all of the following:

- 100BT, 100BF, 1000BT, 1000B-SX, 1000B-LX, 10GBT, 10GB-SR, 10GB-LR, 10GB-ER
- Auto negotiation, MAC layer packets, IP protocols, LLC protocols, Spanning Tree protocols, Quality of Service protocols, ARP, IPV6, IPv4, ICMP, IGMP,

TCP, UDP, IP Security protocols like TLS & SSL, both DHCP & Static Ip Address assignment

- TnMS only allows PTP-V2, master, slave & boundary clocks
- 1 PPS outputs and GPS time support
- Encryption
- SNMP – there is a TmNS management information base (MIB)

This does not imply that all devices must support all of the above, but compared to the other standards it includes far more of the possibilities networked systems have to offer.

TmNS attempts to keep the message format simple by trying to define a single message format to describe all traffic types.

The message definition consists of a message header followed by a message payload. The message payload consist of a series of TmNS packages.

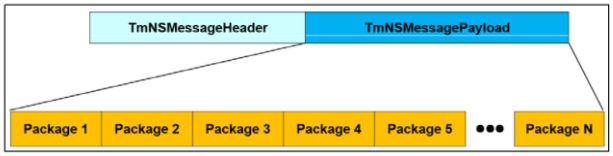

Figure 14: TmNS message format

The TmNS message header consists of message fields very similar to iNET-X.

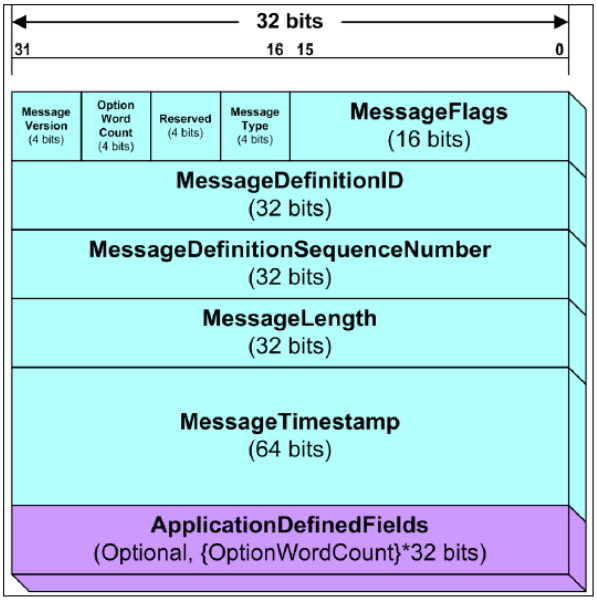

Figure 15 TmNS message header

Each payload package consist of a package header and package payload and must be 32 bit aligned.

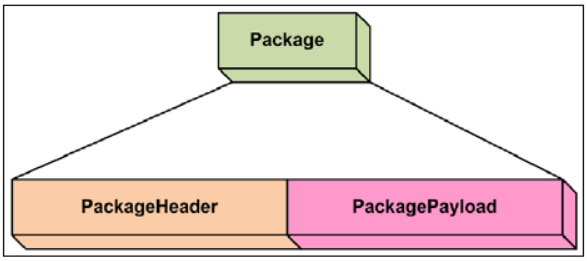

Figure 16: TmNS package

The Package Header contains the fields shown in Figure 17.

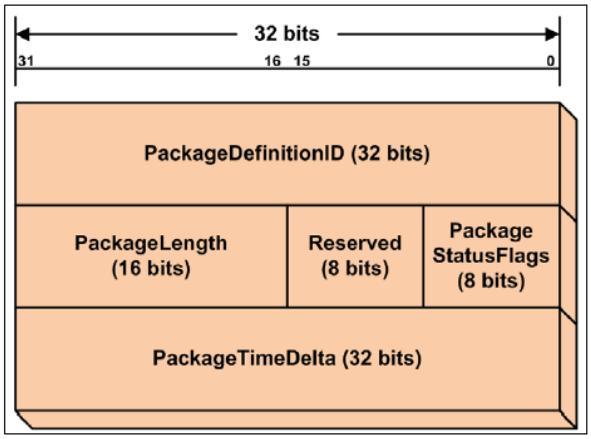

Figure 17: The fields within the package header

While the attempt to use a single package definition is admirable it runs the risk of not being able to describe all data field for all data buses or analog signals inside these restrictions for every vendor. This reduces the possibility of interoperability between multiple vendors.

IRIG-106 chapter 25 requires that TmNS devices support configuration over SNMP, with the use of various public MIBs and a TmNS specific MIB. Again, here it is not clear that all vendor device capabilities can be configured via these MIBs.

TmNS is evolving over time and as adoption increases in the customer base these issues will no doubt be addressed, however, achieving a truly vendor neutral standard via TmNS remains a long way off.

8. Becoming the Babel Fish

From a customer point of view, the benefits of 'plug-and-play' hardware are fairly obvious. The ability to pick and choose between vendors without worrying about integration issues or needing to alter IT infrastructure, has many advantages.

From a vendor point of view, Curtiss-Wright will as always be all things the all people. We will continue to support IENA and INET-X out of Das Studio and DARv3 out of TTCWare. For Chapter 10 our Axon family DAUs will support all time format packets, until the time packet issues in distributed networked systems is addressed by the RCC, all UDP data transfer formats and all data formats, configurable by the user. Both TTCWare and Das Studio will be able to produce TMAT's files at both

DOI 10.5162/ettc2020/2.5

the individual DAU level and at the full system level to support both the single recording per system and single recording per DAU approach.

For TmNS Curtiss-Wright will build in the ability to generate MDL files readable by both Das Studio and TTCWare, but will continue to use our existing configuration software to program the hardware until the MDL schema and the TmNS MIB supports a way to configure all of our hardware settings.