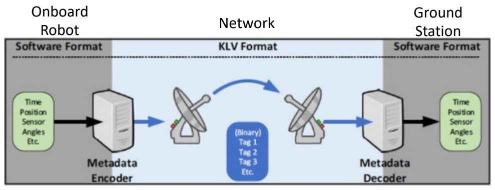
208000459F4A6AA4AA8050271C20602FD

KLV 2 3D

The process of capturing wirelessly networked KLV Local Set data transmitted from an unmanned aerial vehicle (UAS) to visualizing that data in 3D using Open Source, Open Standards technology.

1



PCAPNG File {Wireshark/tshark} → Filter for udp.port==4040 {Wireshark/tshark} → Marked up filtered PCAPNG (Hex Text) (Parse) {pcapng.dfdl.xsd} → file → Extract KLV (UDP payload) (Hex Text) {ExtractPcapng DataValues.xslt} → file → Parse, visualize KLV (Hex Text) as marked up XML {klv.dfdl.xsd} → file

In-memory DOM objects

{DFDL extract KLV Hex}

Java parse KLV telemetry track into DIS

Decode KLV (Hex Binary) {Harder KLV Java} → Decode KLV (Decimal) {West Ridge Systems jMisb} → Generate DIS PDU Tracks {OpenDIS7 Java} → Save/Replay DIS PDU streams {OpenDIS} → Visualize 3D Scene (Playback) {X3D, Spiders3D} → LVC {Plan, Analyze, Train, Archive}

The Blais approach would be KLV XML -> XSLT -> RST XML

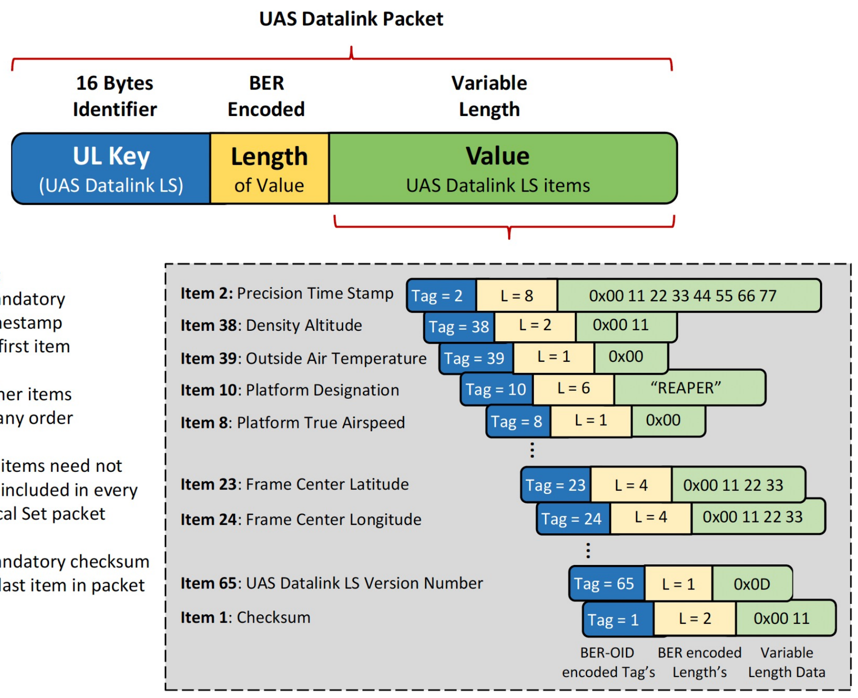
The top right + lower left blocks might be replaced by DIS Schema with DFDL for KLV, as a general pattern for DIS DFDL parsing of any telemetry track values, leapfrog for future telemetry patterns. End result remains DIS XML in either case.

2

KLV

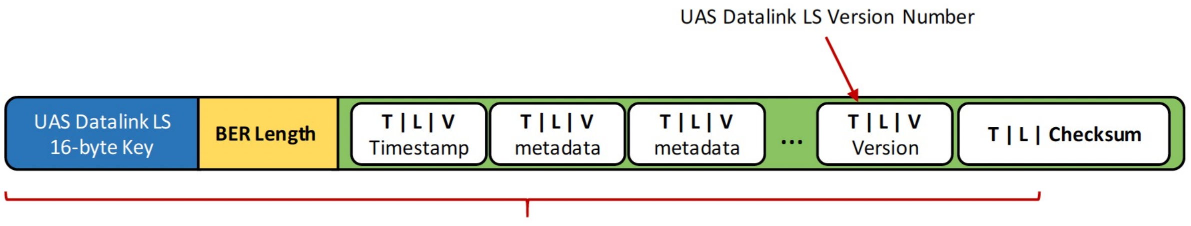
- **What is KLV?**
  - A bandwidth-efficient metadata Local Set conforming to Society of Motion Picture and Television Engineers (SMPTE) ST 336
  - This standard defines a byte-level data encoding protocol for representing data items and data groups. This protocol defines a data structure which is independent of the application or transportation method used
  - https://ieeexplore.ieee.org/document/8019807

3


UAS Datalink Packet
16 Bytes Identifier
BER Encoded
Variable Length
UL Key (UAS Datalink LS)
Length of Value
Value UAS Datalink LS items
Item 2: Precision Time Stamp | Tag = 2 | L = 8 | 0x00 11 22 33 44 55 66 77
Item 38: Density Altitude | Tag = 38 | L = 2 | 0x00 11
Item 39: Outside Air Temperature | Tag = 39 | L = 1 | 0x00
Item 10: Platform Designation | Tag = 10 | L = 6 | "REAPER"
Item 8: Platform True Airspeed | Tag = 8 | L = 1 | 0x00
Item 23: Frame Center Latitude | Tag = 23 | L = 4 | 0x00 11 22 33
Item 24: Frame Center Longitude | Tag = 24 | L = 4 | 0x00 11 22 33
Item 65: UAS Datalink LS Version Number | Tag = 65 | L = 1 | 0x0D
Item 1: Checksum | Tag = 1 | L = 2 | 0x00 11
BER-OID encoded Tag's
BER encoded Length's
Variable Length Data

Notes:
- Mandatory timestamp as first item
- Other items in any order
- All items need not be included in every Local Set packet
- Mandatory checksum as last item in packet

4



Checksum is computed from the start of the 16-byte Key up to and including the 1-byte length field in the Checksum metadata item.

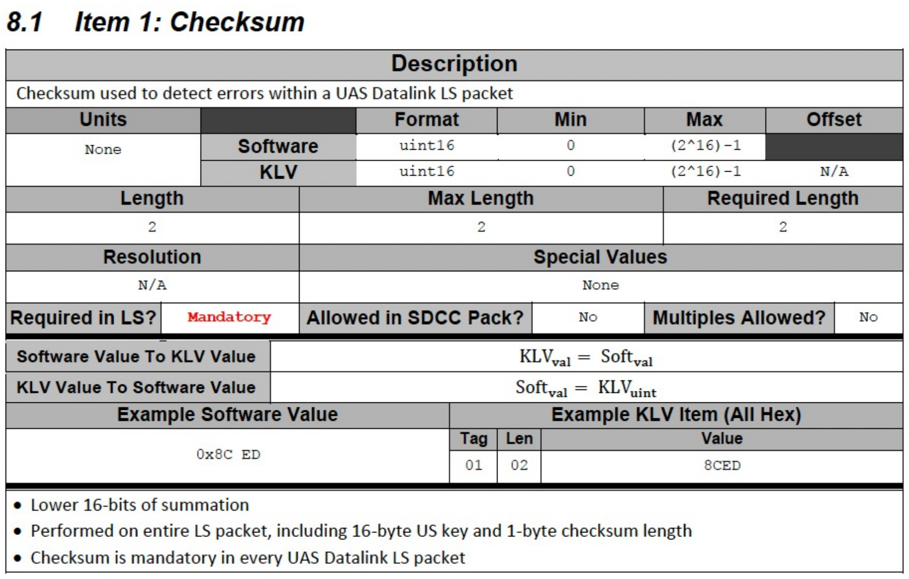
5

ST 0601.17 UAS Datalink Local Set

Tag	Name	Units	Format	Len	SDCC	MUL	Description
2	Precision Time Stamp	µs	uint64	8	N	N	Timestamp for all metadata in this Local Set; used to coordinate with Motion Imagery
3	Mission ID	None	utf8	V	N	N	Descriptive mission identifier to distinguish event or sortie
4	Platform Tail Number	None	utf8	V	N	N	Identifier of platform as posted
5	Platform Heading Angle	°	uint16	2	Y	N	Aircraft heading angle
6	Platform Pitch Angle	°	int16	2	Y	N	Aircraft pitch angle
7	Platform Roll Angle	°	int16	2	Y	N	Platform roll angle
8	Platform True Airspeed	m/s	uint8	1	Y	N	True airspeed (TAS) of platform
9	Platform Indicated Airspeed	m/s	uint8	1	Y	N	Indicated airspeed (IAS) of platform
10	Platform Designation	None	utf8	V	N	N	Model name for the platform
11	Image Source Sensor	None	utf8	V	N	N	Name of currently active sensor
12	Image Coordinate System	None	utf8	V	N	N	Name of the image coordinate system used
13	Sensor Latitude	°	int32	4	Y	N	Sensor latitude
14	Sensor Longitude	°	int32	4	Y	N	Sensor longitude
15	Sensor True Altitude	m	uint16	2	Y	N	Altitude of sensor as measured from Mean Sea Level (MSL)

Currently, 142 KLV tags

6



8.1 Item 1: Checksum

Description					
Checksum used to detect errors within a UAS Datalink LS packet					
Units		**Format**	**Min**	**Max**	**Offset**
None	**Software**	uint16	0	(2^16)-1	
	KLV	uint16	0	(2^16)-1	N/A

Length	Max Length	Required Length
2	2	2

Resolution	Special Values
N/A	None

Required in LS?	Mandatory	Allowed in SDCC Pack?	No	Multiples Allowed?	No

Software Value To KLV Value	$KLV_{val} = Soft_{val}$
KLV Value To Software Value	$Soft_{val} = KLV_{uint}$

Example Software Value	Example KLV Item (All Hex)		
	Tag	Len	Value
0x8C ED	01	02	8CED

- Lower 16-bits of summation
- Performed on entire LS packet, including 16-byte US key and 1-byte checksum length
- Checksum is mandatory in every UAS Datalink LS packet

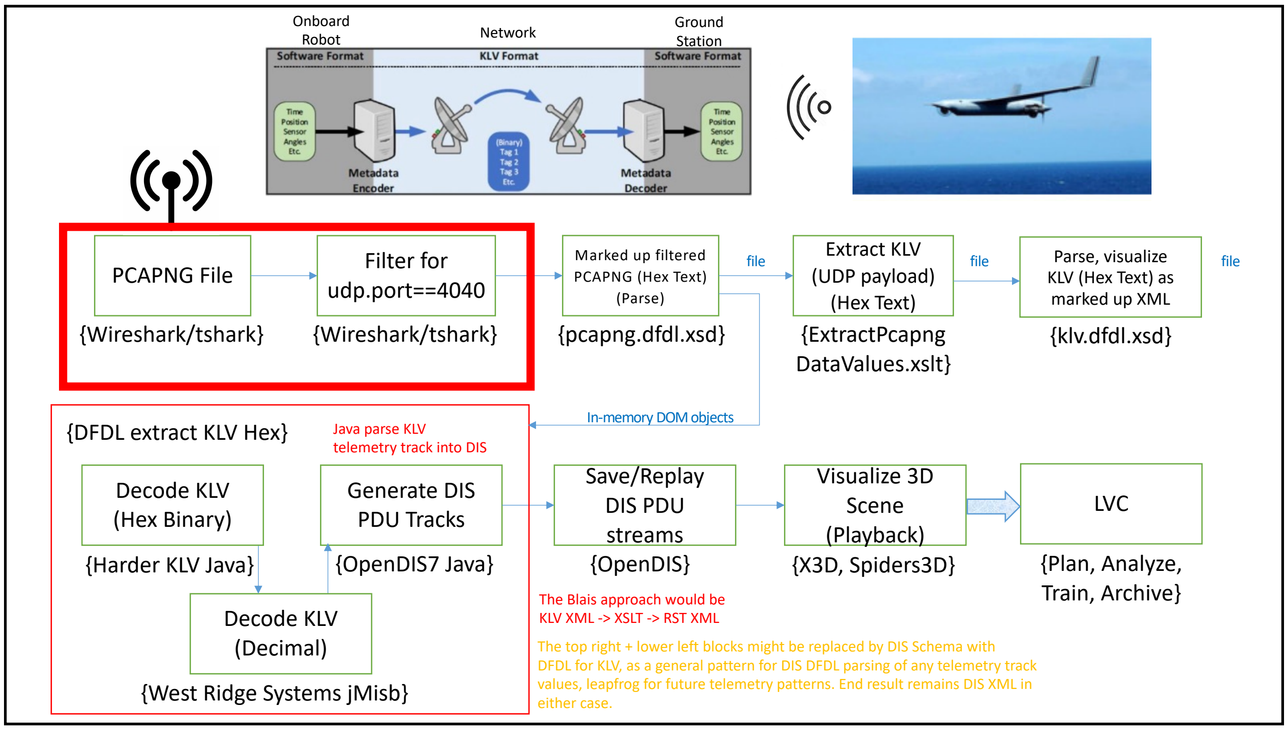
7

KLV in HEX

```
060E2B34020B01010E01030101000000690208000459F4A6AA4AA8050271C20602FD3D070208B8
0A054D51312D420B02454F0D045595B66D0E045B5360C40F02C2211002CD9C1102D9171204724
A0A20130487F84B8614047DC55ECE1504038309261602128117O4F101A229180414BC082B190234F30102EF00
```

UAS Local Set Universal Label

8


Onboard Robot
Network
Ground Station
Software Format
KLV Format
Software Format
Time Position Sensor Angles Etc.
(Binary) Tag 1 Tag 2 Tag 3 Etc.
Metadata Encoder
Metadata Decoder
PCAPNG File
{Wireshark/tshark}
Filter for udp.port==4040
{Wireshark/tshark}
Marked up filtered PCAPNG (Hex Text) (Parse)
{pcapng.dfdl.xsd}
file
Extract KLV (UDP payload) (Hex Text)
{ExtractPcapng DataValues.xslt}
file
Parse, visualize KLV (Hex Text) as marked up XML
{klv.dfdl.xsd}
file
In-memory DOM objects
{DFDL extract KLV Hex}
Java parse KLV telemetry track into DIS
Decode KLV (Hex Binary)
{Harder KLV Java}
Generate DIS PDU Tracks
{OpenDIS7 Java}
Decode KLV (Decimal)
{West Ridge Systems jMisb}
Save/Replay DIS PDU streams
{OpenDIS}
Visualize 3D Scene (Playback)
{X3D, Spiders3D}
LVC
{Plan, Analyze, Train, Archive}
The Blais approach would be KLV XML -> XSLT -> RST XML
The top right + lower left blocks might be replaced by DIS Schema with DFDL for KLV, as a general pattern for DIS DFDL parsing of any telemetry track values, leapfrog for future telemetry patterns. End result remains DIS XML in either case.

9

Wireshark

- **What is Wireshark?**
  - Wireshark is the world's most popular network protocol analyzer. It is used for troubleshooting, analysis, development and education.
  - https://www.wireshark.org/docs/relnotes/wireshark-3.4.9.html

  - Used to capture an Unmanned Aerial System (UAS) Datalink Local Set from an MPEG-2 transport container along with compressed Motion Imagery over a wireless network

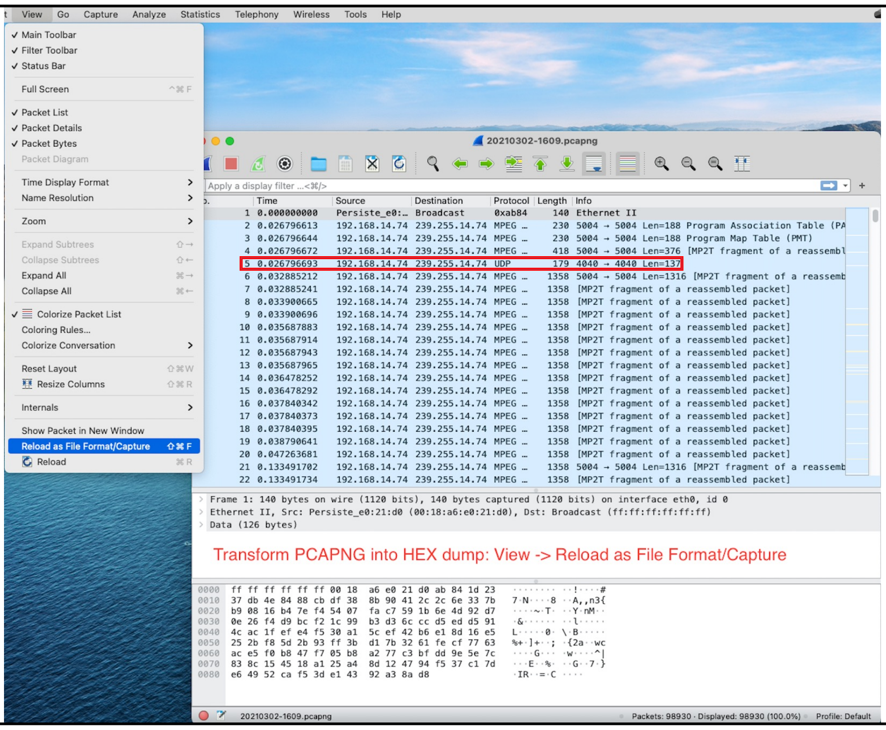
10

PCAPNG

- **What is PCAPNG?**
  - The PCAP **Next Generation Dump File** Format (or pcapng for short) is an attempt to overcome the limitations of the currently widely used (but limited) libpcap format.
- **Excerpt from the Network Working Group on PCAPNG**
  - The problem of exchanging packet traces becomes more and more critical every day; unfortunately, no standard solutions exist for this task right now. One of the most accepted packet interchange formats is the one defined by libpcap, which is rather old and is lacking in functionality for more modern applications particularly from the extensibility point of view.
  - https://pcapng.github.io/pcapng/draft-tuexen-opsawg-pcapng.html

11

MPEG-2 Transport Container Interspersed with UDP Packets



```
 o.  Time         Source          Destination     Protocol Length Info
   1 0.000000000  Persiste_e0:…   Broadcast       0xab84     140  Ethernet II
   2 0.026796613  192.168.14.74   239.255.14.74   MPEG …     230  5004 → 5004 Len=188 Program Association Table (PA
   3 0.026796644  192.168.14.74   239.255.14.74   MPEG …     230  5004 → 5004 Len=188 Program Map Table (PMT)
   4 0.026796672  192.168.14.74   239.255.14.74   MPEG …     418  5004 → 5004 Len=376 [MP2T fragment of a reassembl
   5 0.026796693  192.168.14.74   239.255.14.74   UDP        179  4040 → 4040 Len=137
   6 0.032885212  192.168.14.74   239.255.14.74   MPEG …    1358  5004 → 5004 Len=1316 [MP2T fragment of a reassemb
   7 0.032885241  192.168.14.74   239.255.14.74   MPEG …    1358  [MP2T fragment of a reassembled packet]
   8 0.033900665  192.168.14.74   239.255.14.74   MPEG …    1358  [MP2T fragment of a reassembled packet]
   9 0.033900696  192.168.14.74   239.255.14.74   MPEG …    1358  [MP2T fragment of a reassembled packet]
  10 0.035687883  192.168.14.74   239.255.14.74   MPEG …    1358  [MP2T fragment of a reassembled packet]
  11 0.035687914  192.168.14.74   239.255.14.74   MPEG …    1358  [MP2T fragment of a reassembled packet]
  12 0.035687943  192.168.14.74   239.255.14.74   MPEG …    1358  [MP2T fragment of a reassembled packet]
  13 0.035687965  192.168.14.74   239.255.14.74   MPEG …    1358  [MP2T fragment of a reassembled packet]
  14 0.036478252  192.168.14.74   239.255.14.74   MPEG …    1358  [MP2T fragment of a reassembled packet]
  15 0.036478292  192.168.14.74   239.255.14.74   MPEG …    1358  [MP2T fragment of a reassembled packet]
  16 0.037840342  192.168.14.74   239.255.14.74   MPEG …    1358  [MP2T fragment of a reassembled packet]
  17 0.037840373  192.168.14.74   239.255.14.74   MPEG …    1358  [MP2T fragment of a reassembled packet]
  18 0.037840395  192.168.14.74   239.255.14.74   MPEG …    1358  [MP2T fragment of a reassembled packet]
  19 0.038790641  192.168.14.74   239.255.14.74   MPEG …    1358  [MP2T fragment of a reassembled packet]
  20 0.047263681  192.168.14.74   239.255.14.74   MPEG …    1358  [MP2T fragment of a reassembled packet]
  21 0.133491702  192.168.14.74   239.255.14.74   MPEG …    1358  5004 → 5004 Len=1316 [MP2T fragment of a reassemb
  22 0.133491734  192.168.14.74   239.255.14.74   MPEG …    1358  [MP2T fragment of a reassembled packet]

> Frame 1: 140 bytes on wire (1120 bits), 140 bytes captured (1120 bits) on interface eth0, id 0
> Ethernet II, Src: Persiste_e0:21:d0 (00:18:a6:e0:21:d0), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
> Data (126 bytes)
```

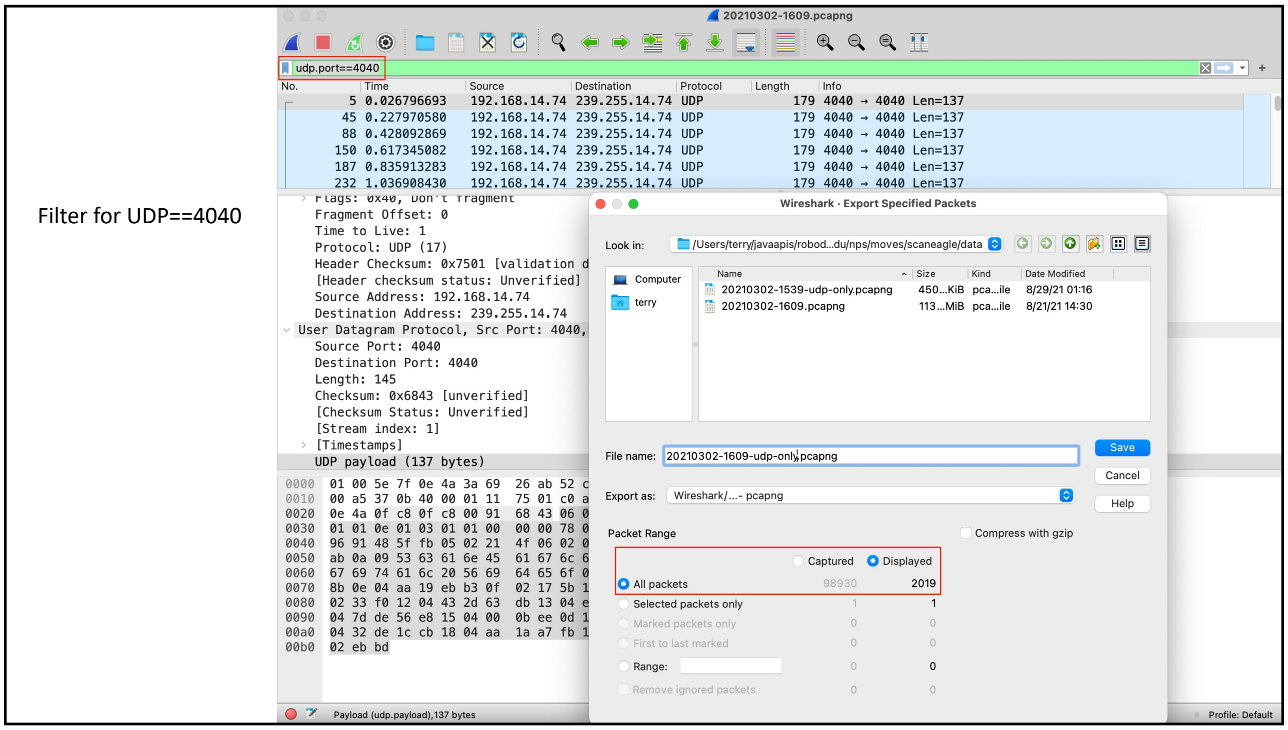
Transform PCAPNG into HEX dump: View -> Reload as File Format/Capture

```
0000  ff ff ff ff ff ff 00 18  a6 e0 21 d0 ab 84 1d 23   ········ ··!····#
0010  37 db 4e 84 88 cb df 38  8b 90 41 2c 2c 6e 33 7b   7·N····8 ··A,,n3{
0020  b9 08 16 b4 7e f4 54 07  fa c7 59 1b 6e 4d 92 d7   ····~·T· ··Y·nM··
0030  0e 26 f4 d9 bc f2 1c 99  b3 d3 6c cc d5 ed d5 91   ·&······ ··l·····
0040  4c ac 1f ef e4 f5 30 a1  5c ef 42 b6 e1 8d 16 e5   L·····0· \·B·····
0050  25 2b f8 5d 2b 93 ff 3b  d1 7b 32 61 fe cf 77 63   %+·]+··; ·{2a··wc
0060  ac e5 f0 b8 47 f7 05 b8  a2 77 c3 bf dd 9e 5e 7c   ····G··· ·w····^|
0070  83 8c 15 45 18 a1 25 a4  8d 12 47 94 f5 37 c1 7d   ···E··%· ··G··7·}
0080  e6 49 52 ca f5 3d e1 43  92 a3 8a d8               ·IR··=·C ····
```

20210302-1609.pcapng   Packets: 98930 · Displayed: 98930 (100.0%)   Profile: Default

12

Filter for UDP==4040



20210302-1609.pcapng

udp.port==4040

```
No.    Time         Source        Destination   Protocol  Length Info
     5 0.026796693  192.168.14.74 239.255.14.74 UDP          179 4040 → 4040 Len=137
    45 0.227970580  192.168.14.74 239.255.14.74 UDP          179 4040 → 4040 Len=137
    88 0.428092869  192.168.14.74 239.255.14.74 UDP          179 4040 → 4040 Len=137
   150 0.617345082  192.168.14.74 239.255.14.74 UDP          179 4040 → 4040 Len=137
   187 0.835913283  192.168.14.74 239.255.14.74 UDP          179 4040 → 4040 Len=137
   232 1.036908430  192.168.14.74 239.255.14.74 UDP          179 4040 → 4040 Len=137
```

```
  > Flags: 0x40, Don't fragment
    Fragment Offset: 0
    Time to Live: 1
    Protocol: UDP (17)
    Header Checksum: 0x7501 [validation d
    [Header checksum status: Unverified]
    Source Address: 192.168.14.74
    Destination Address: 239.255.14.74
v User Datagram Protocol, Src Port: 4040,
    Source Port: 4040
    Destination Port: 4040
    Length: 145
    Checksum: 0x6843 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 1]
  > [Timestamps]
    UDP payload (137 bytes)
```

```
0000  01 00 5e 7f 0e 4a 3a 69  26 ab 52 c
0010  00 a5 37 0b 40 00 01 11  75 01 c0 a
0020  0e 4a 0f c8 0f c8 00 91  68 43 06 0
0030  01 01 0e 01 03 01 01 00  00 00 78 0
0040  96 91 48 5f fb 05 02 21  4f 06 02 0
0050  ab 0a 09 53 63 61 6e 45  61 67 6c 6
0060  67 69 74 61 6c 20 56 69  64 65 6f 0
0070  8b 0e 04 aa 19 eb b3 0f  02 17 5b 1
0080  02 33 f0 12 04 43 2d 63  db 13 04 e
0090  04 7d de 56 e8 15 04 00  0b ee 0d 1
00a0  04 32 de 1c cb 18 04 aa  1a a7 fb 1
00b0  02 eb bd
```

Payload (udp.payload),137 bytes

Profile: Default

Wireshark · Export Specified Packets

Look in: /Users/terry/javaapis/robod...du/nps/moves/scaneagle/data

Computer
terry

Name	Size	Kind	Date Modified
20210302-1539-udp-only.pcapng	450...KiB	pca...ile	8/29/21 01:16
20210302-1609.pcapng	113...MiB	pca...ile	8/21/21 14:30

File name: 20210302-1609-udp-only.pcapng

Export as: Wireshark/...- pcapng

Save
Cancel
Help

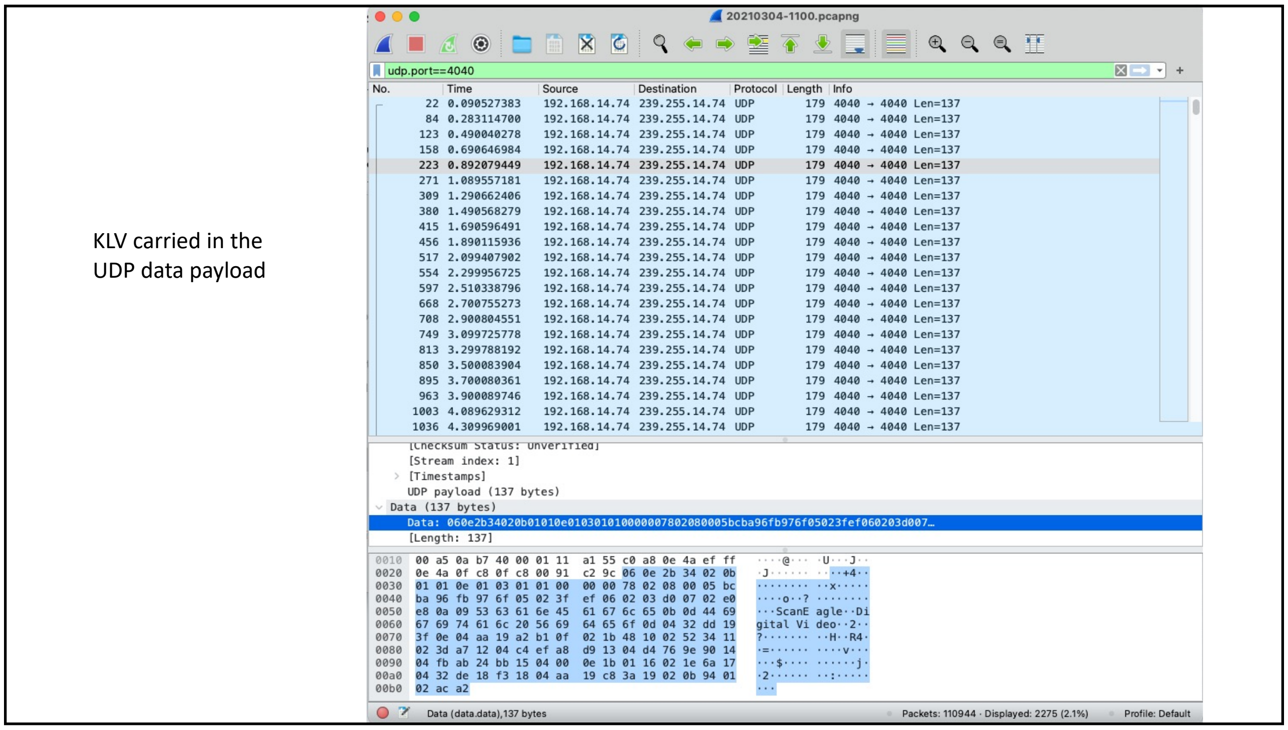
Packet Range

Compress with gzip

	Captured	[✓ Displayed]
[✓ All packets]	98930	2019
Selected packets only	1	1
Marked packets only	0	0
First to last marked	0	0
Range:	0	0
Remove ignored packets	0	0

13

KLV carried in the UDP data payload



20210304-1100.pcapng

udp.port==4040

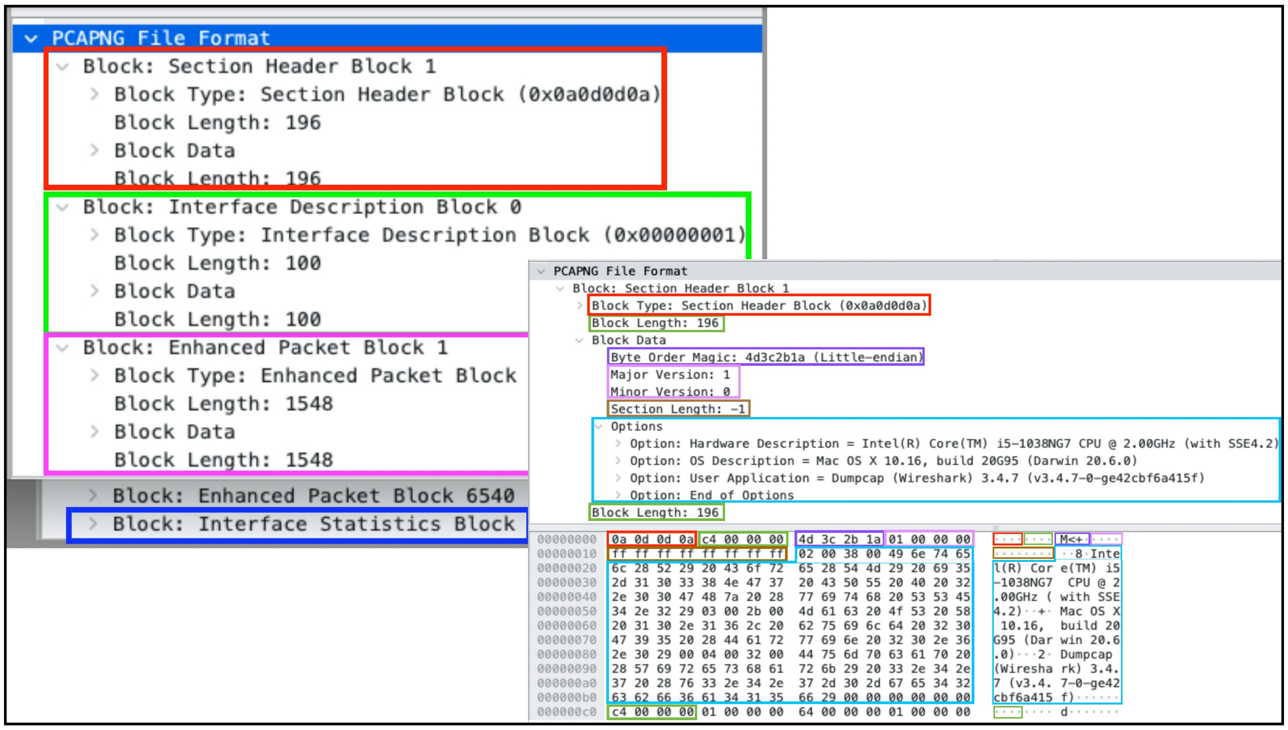
```
No.   Time         Source        Destination   Protocol Length Info
   22 0.090527383  192.168.14.74 239.255.14.74 UDP         179 4040 → 4040 Len=137
   84 0.283114700  192.168.14.74 239.255.14.74 UDP         179 4040 → 4040 Len=137
  123 0.490040278  192.168.14.74 239.255.14.74 UDP         179 4040 → 4040 Len=137
  158 0.690646984  192.168.14.74 239.255.14.74 UDP         179 4040 → 4040 Len=137
  223 0.892079449  192.168.14.74 239.255.14.74 UDP         179 4040 → 4040 Len=137
  271 1.089557181  192.168.14.74 239.255.14.74 UDP         179 4040 → 4040 Len=137
  309 1.290662406  192.168.14.74 239.255.14.74 UDP         179 4040 → 4040 Len=137
  380 1.490568279  192.168.14.74 239.255.14.74 UDP         179 4040 → 4040 Len=137
  415 1.690596491  192.168.14.74 239.255.14.74 UDP         179 4040 → 4040 Len=137
  456 1.890115936  192.168.14.74 239.255.14.74 UDP         179 4040 → 4040 Len=137
  517 2.099407902  192.168.14.74 239.255.14.74 UDP         179 4040 → 4040 Len=137
  554 2.299956725  192.168.14.74 239.255.14.74 UDP         179 4040 → 4040 Len=137
  597 2.510338796  192.168.14.74 239.255.14.74 UDP         179 4040 → 4040 Len=137
  668 2.700755273  192.168.14.74 239.255.14.74 UDP         179 4040 → 4040 Len=137
  708 2.900804551  192.168.14.74 239.255.14.74 UDP         179 4040 → 4040 Len=137
  749 3.099725778  192.168.14.74 239.255.14.74 UDP         179 4040 → 4040 Len=137
  813 3.299788192  192.168.14.74 239.255.14.74 UDP         179 4040 → 4040 Len=137
  850 3.500083904  192.168.14.74 239.255.14.74 UDP         179 4040 → 4040 Len=137
  895 3.700080361  192.168.14.74 239.255.14.74 UDP         179 4040 → 4040 Len=137
  963 3.900089746  192.168.14.74 239.255.14.74 UDP         179 4040 → 4040 Len=137
 1003 4.089629312  192.168.14.74 239.255.14.74 UDP         179 4040 → 4040 Len=137
 1036 4.309969001  192.168.14.74 239.255.14.74 UDP         179 4040 → 4040 Len=137
```

```
    [Checksum Status: Unverified]
    [Stream index: 1]
  > [Timestamps]
    UDP payload (137 bytes)
v Data (137 bytes)
    Data: 060e2b34020b01010e010301010000007802080005bcba96fb976f05023fef060203d007…
    [Length: 137]
```

```
0010  00 a5 0a b7 40 00 01 11  a1 55 c0 a8 0e 4a ef ff   ····@··· ·U···J··
0020  0e 4a 0f c8 0f c8 00 91  c2 9c 06 0e 2b 34 02 0b   ·J······ ····+4··
0030  01 01 0e 01 03 01 01 00  00 00 78 02 08 00 05 bc   ········ ··x·····
0040  ba 96 fb 97 6f 05 02 3f  ef 06 02 03 d0 07 02 e0   ····o··? ········
0050  e8 0a 09 53 63 61 6e 45  61 67 6c 65 0b 0d 44 69   ···ScanE agle··Di
0060  67 69 74 61 6c 20 56 69  64 65 6f 0d 04 32 dd 19   gital Vi deo··2··
0070  3f 0e 04 aa 19 a2 b1 0f  02 1b 48 10 02 52 34 11   ?······· ··H··R4·
0080  02 3d a7 12 04 c4 ef a8  d9 13 04 d4 76 9e 90 14   ·=······ ····v···
0090  04 fb ab 24 bb 15 04 00  0e 1b 01 16 02 1e 6a 17   ···$···· ······j·
00a0  04 32 de 18 f3 18 04 aa  19 c8 3a 19 02 0b 94 01   ·2······ ··:·····
00b0  02 ac a2                                           ···
```

Data (data.data),137 bytes   Packets: 110944 · Displayed: 2275 (2.1%)   Profile: Default

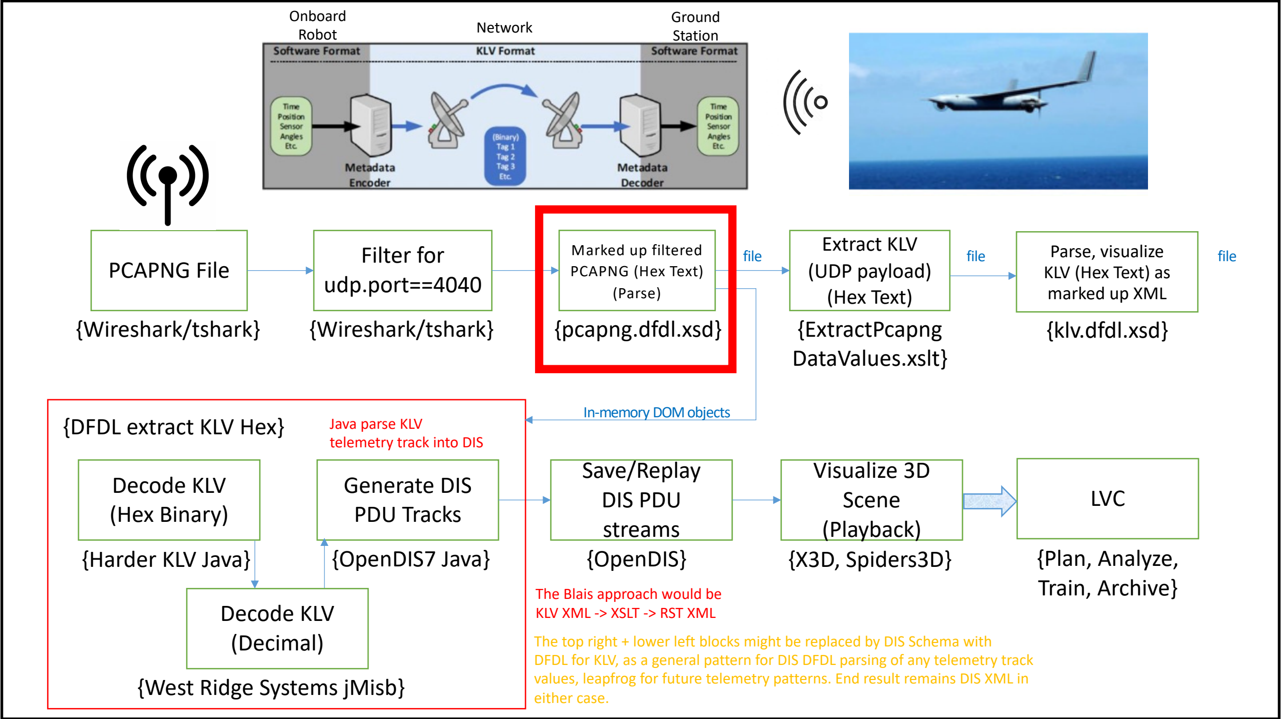
14



```
PCAPNG File Format
  Block: Section Header Block 1
    > Block Type: Section Header Block (0x0a0d0d0a)
      Block Length: 196
    > Block Data
      Block Length: 196
  Block: Interface Description Block 0
    > Block Type: Interface Description Block (0x00000001)
      Block Length: 100
    > Block Data
      Block Length: 100
  Block: Enhanced Packet Block 1
    > Block Type: Enhanced Packet Block
      Block Length: 1548
    > Block Data
      Block Length: 1548
    > Block: Enhanced Packet Block 6540
    > Block: Interface Statistics Block
```

```
PCAPNG File Format
  Block: Section Header Block 1
    > Block Type: Section Header Block (0x0a0d0d0a)
      Block Length: 196
      Block Data
        Byte Order Magic: 4d3c2b1a (Little-endian)
        Major Version: 1
        Minor Version: 0
        Section Length: -1
        Options
          > Option: Hardware Description = Intel(R) Core(TM) i5-1038NG7 CPU @ 2.00GHz (with SSE4.2)
          > Option: OS Description = Mac OS X 10.16, build 20G95 (Darwin 20.6.0)
          > Option: User Application = Dumpcap (Wireshark) 3.4.7 (v3.4.7-0-ge42cbf6a415f)
          > Option: End of Options
      Block Length: 196
```

```
00000000  0a 0d 0d 0a c4 00 00 00  4d 3c 2b 1a 01 00 00 00   ········M<+·····
00000010  ff ff ff ff ff ff ff ff  02 00 38 00 49 6e 74 65   ··········8·Inte
00000020  6c 28 52 29 20 43 6f 72  65 28 54 4d 29 20 69 35   l(R) Cor e(TM) i5
00000030  2d 31 30 33 38 4e 47 37  20 43 50 55 20 40 20 32   -1038NG7  CPU @ 2
00000040  2e 30 30 47 48 7a 20 28  77 69 74 68 20 53 53 45   .00GHz ( with SSE
00000050  34 2e 32 29 03 00 2b 00  4d 61 63 20 4f 53 20 58   4.2)··+· Mac OS X
00000060  20 31 30 2e 31 36 2c 20  62 75 69 6c 64 20 32 30    10.16,  build 20
00000070  47 39 35 20 28 44 61 72  77 69 6e 20 32 30 2e 36   G95 (Dar win 20.6
00000080  2e 30 29 00 04 00 32 00  44 75 6d 70 63 61 70 20   .0)···2· Dumpcap
00000090  28 57 69 72 65 73 68 61  72 6b 29 20 33 2e 34 2e   (Wiresha rk) 3.4.
000000a0  37 20 28 76 33 2e 34 2e  37 2d 30 2d 67 65 34 32   7 (v3.4. 7-0-ge42
000000b0  63 62 66 36 61 34 31 35  66 29 00 00 00 00 00 00   cbf6a415 f)······
000000c0  c4 00 00 00 01 00 00 00  64 00 00 00 01 00 00 00   ········ d·······
```

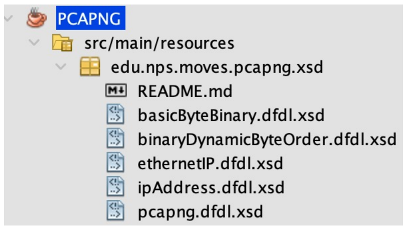

Onboard Robot
Network
Ground Station
Software Format
KLV Format
Software Format
Time Position Sensor Angles Etc.
Metadata Encoder
(Binary) Tag 1 Tag 2 Tag 3 Etc.
Metadata Decoder
Time Position Sensor Angles Etc.
PCAPNG File
{Wireshark/tshark}
Filter for udp.port==4040
{Wireshark/tshark}
Marked up filtered PCAPNG (Hex Text) (Parse)
{pcapng.dfdl.xsd}
file
Extract KLV (UDP payload) (Hex Text)
{ExtractPcapng DataValues.xslt}
file
Parse, visualize KLV (Hex Text) as marked up XML
{klv.dfdl.xsd}
file
In-memory DOM objects
{DFDL extract KLV Hex}
Java parse KLV telemetry track into DIS
Decode KLV (Hex Binary)
{Harder KLV Java}
Generate DIS PDU Tracks
{OpenDIS7 Java}
Decode KLV (Decimal)
{West Ridge Systems jMisb}
Save/Replay DIS PDU streams
{OpenDIS}
Visualize 3D Scene (Playback)
{X3D, Spiders3D}
LVC
{Plan, Analyze, Train, Archive}
The Blais approach would be KLV XML -> XSLT -> RST XML
The top right + lower left blocks might be replaced by DIS Schema with DFDL for KLV, as a general pattern for DIS DFDL parsing of any telemetry track values, leapfrog for future telemetry patterns. End result remains DIS XML in either case.

Apache Daffodil

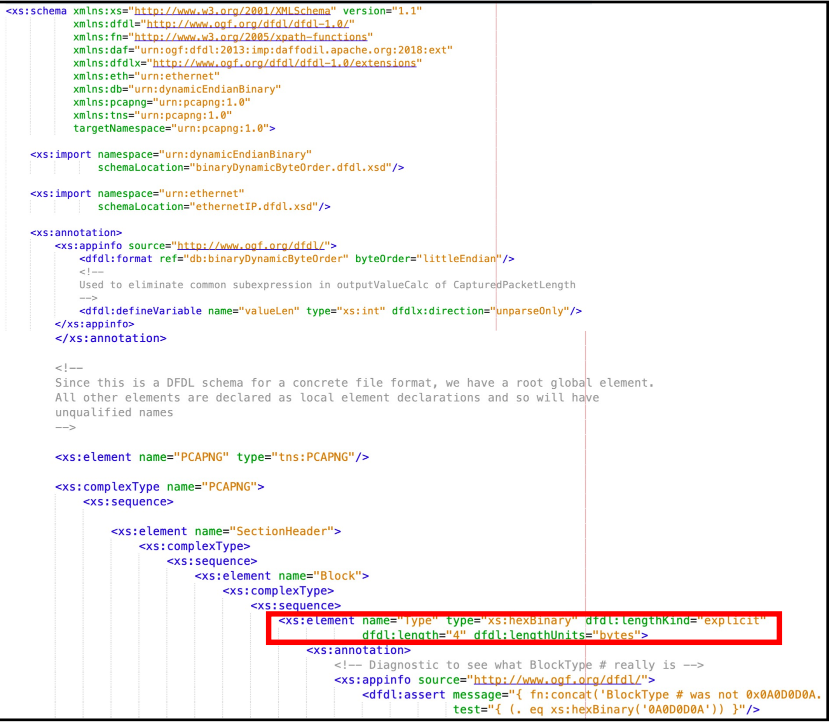
- What is Apache Daffodil?
  - Open-source implementation of the Data Format Description Language to convert between fixed format data and XML, JSON, and other data structures.
  - The Data Format Description Language (DFDL) is a specification, developed by the Open Grid Forum, capable of describing many data formats, including both textual and binary, scientific and numeric, legacy and modern, commercial record-oriented, and many industry and military standards. It defines a language that is a subset of W3C XML schema to describe the logical format of the data, and annotations within the schema to describe the physical representation.

17

DFDL decorated
XML Schema
for PCAPNG



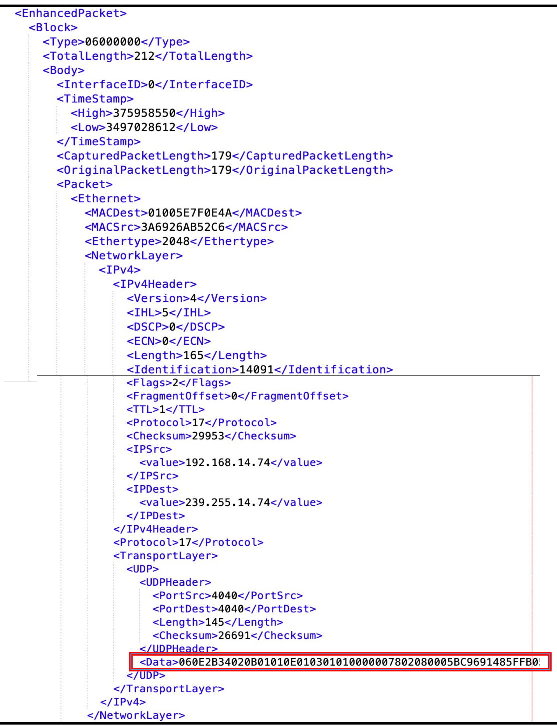
```
PCAPNG
  src/main/resources
    edu.nps.moves.pcapng.xsd
      README.md
      basicByteBinary.dfdl.xsd
      binaryDynamicByteOrder.dfdl.xsd
      ethernetIP.dfdl.xsd
      ipAddress.dfdl.xsd
      pcapng.dfdl.xsd
```



```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" version="1.1"
           xmlns:dfdl="http://www.ogf.org/dfdl/dfdl-1.0/"
           xmlns:fn="http://www.w3.org/2005/xpath-functions"
           xmlns:daf="urn:ogf:dfdl:2013:imp:daffodil.apache.org:2018:ext"
           xmlns:dfdlx="http://www.ogf.org/dfdl/dfdl-1.0/extensions"
           xmlns:eth="urn:ethernet"
           xmlns:db="urn:dynamicEndianBinary"
           xmlns:pcapng="urn:pcapng:1.0"
           xmlns:tns="urn:pcapng:1.0"
           targetNamespace="urn:pcapng:1.0">

    <xs:import namespace="urn:dynamicEndianBinary"
               schemaLocation="binaryDynamicByteOrder.dfdl.xsd"/>

    <xs:import namespace="urn:ethernet"
               schemaLocation="ethernetIP.dfdl.xsd"/>

    <xs:annotation>
        <xs:appinfo source="http://www.ogf.org/dfdl/">
            <dfdl:format ref="db:binaryDynamicByteOrder" byteOrder="littleEndian"/>
            <!--
            Used to eliminate common subexpression in outputValueCalc of CapturedPacketLength
            -->
            <dfdl:defineVariable name="valueLen" type="xs:int" dfdlx:direction="unparseOnly"/>
        </xs:appinfo>
        </xs:annotation>

        <!--
        Since this is a DFDL schema for a concrete file format, we have a root global element.
        All other elements are declared as local element declarations and so will have
        unqualified names
        -->

        <xs:element name="PCAPNG" type="tns:PCAPNG"/>

        <xs:complexType name="PCAPNG">
            <xs:sequence>

                <xs:element name="SectionHeader">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="Block">
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element name="Type" type="xs:hexBinary" dfdl:lengthKind="explicit"
                                                    dfdl:length="4" dfdl:lengthUnits="bytes">
                                            <xs:annotation>
                                                <!-- Diagnostic to see what BlockType # really is -->
                                                <xs:appinfo source="http://www.ogf.org/dfdl/">
                                                    <dfdl:assert message="{ fn:concat('BlockType # was not 0x0A0D0D0A.
                                                                 test="{ (. eq xs:hexBinary('0A0D0D0A')) }"/>
```

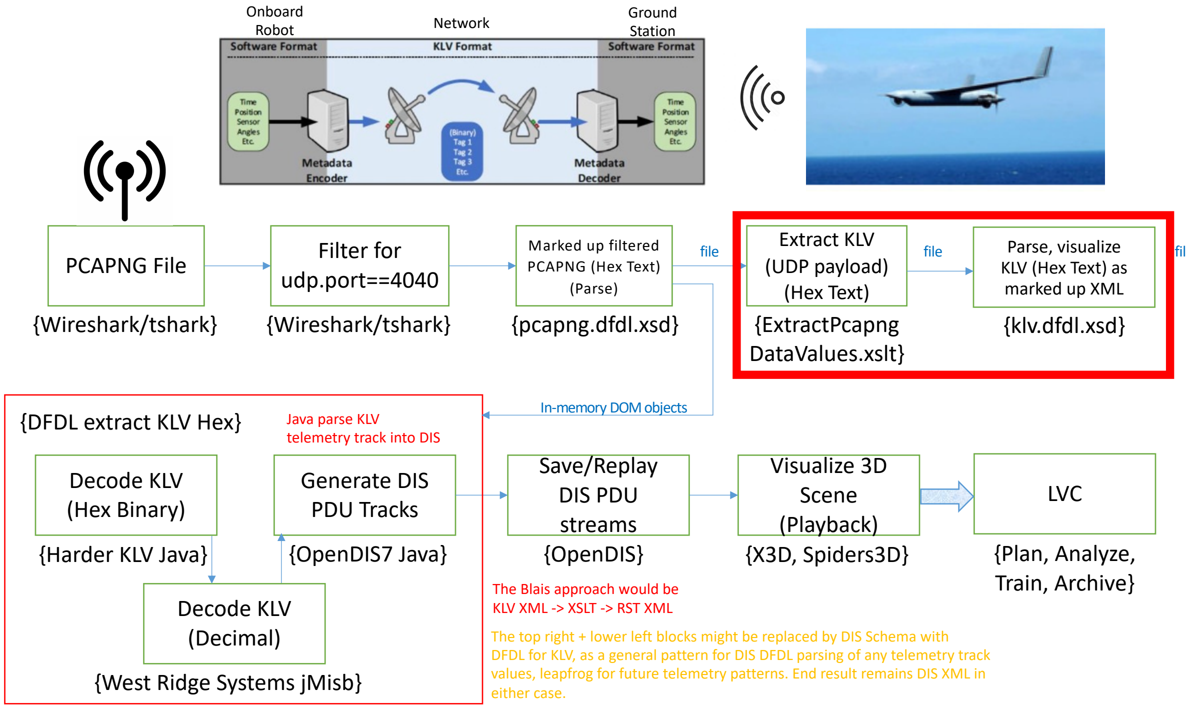
18

Parsed PCAPNG to XML Format



```
<EnhancedPacket>
  <Block>
    <Type>06000000</Type>
    <TotalLength>212</TotalLength>
    <Body>
      <InterfaceID>0</InterfaceID>
      <TimeStamp>
        <High>375958550</High>
        <Low>3497028612</Low>
      </TimeStamp>
      <CapturedPacketLength>179</CapturedPacketLength>
      <OriginalPacketLength>179</OriginalPacketLength>
      <Packet>
        <Ethernet>
          <MACDest>01005E7F0E4A</MACDest>
          <MACSrc>3A6926AB52C6</MACSrc>
          <Ethertype>2048</Ethertype>
          <NetworkLayer>
            <IPv4>
              <IPv4Header>
                <Version>4</Version>
                <IHL>5</IHL>
                <DSCP>0</DSCP>
                <ECN>0</ECN>
                <Length>165</Length>
                <Identification>14091</Identification>
                <Flags>2</Flags>
                <FragmentOffset>0</FragmentOffset>
                <TTL>1</TTL>
                <Protocol>17</Protocol>
                <Checksum>29953</Checksum>
                <IPSrc>
                  <value>192.168.14.74</value>
                </IPSrc>
                <IPDest>
                  <value>239.255.14.74</value>
                </IPDest>
              </IPv4Header>
              <Protocol>17</Protocol>
              <TransportLayer>
                <UDP>
                  <UDPHeader>
                    <PortSrc>4040</PortSrc>
                    <PortDest>4040</PortDest>
                    <Length>145</Length>
                    <Checksum>26691</Checksum>
                  </UDPHeader>
                  <Data>060E2B34020B01010E0103010100000078020800005BC9691485FFB0
                </UDP>
              </TransportLayer>
            </IPv4>
          </NetworkLayer>
```

19



Onboard Robot
Network
Ground Station
Software Format
KLV Format
Software Format
Time Position Sensor Angles Etc.
Metadata Encoder
(Binary) Tag 1 Tag 2 Tag 3 Etc.
Metadata Decoder
Time Position Sensor Angles Etc.
PCAPNG File
{Wireshark/tshark}
Filter for udp.port==4040
{Wireshark/tshark}
Marked up filtered PCAPNG (Hex Text) (Parse)
{pcapng.dfdl.xsd}
file
Extract KLV (UDP payload) (Hex Text)
{ExtractPcapng DataValues.xslt}
file
Parse, visualize KLV (Hex Text) as marked up XML
{klv.dfdl.xsd}
file
In-memory DOM objects
{DFDL extract KLV Hex}
Java parse KLV telemetry track into DIS
Decode KLV (Hex Binary)
{Harder KLV Java}
Generate DIS PDU Tracks
{OpenDIS7 Java}
Decode KLV (Decimal)
{West Ridge Systems jMisb}
Save/Replay DIS PDU streams
{OpenDIS}
Visualize 3D Scene (Playback)
{X3D, Spiders3D}
LVC
{Plan, Analyze, Train, Archive}

The Blais approach would be KLV XML -> XSLT -> RST XML

The top right + lower left blocks might be replaced by DIS Schema with DFDL for KLV, as a general pattern for DIS DFDL parsing of any telemetry track values, leapfrog for future telemetry patterns. End result remains DIS XML in either case.

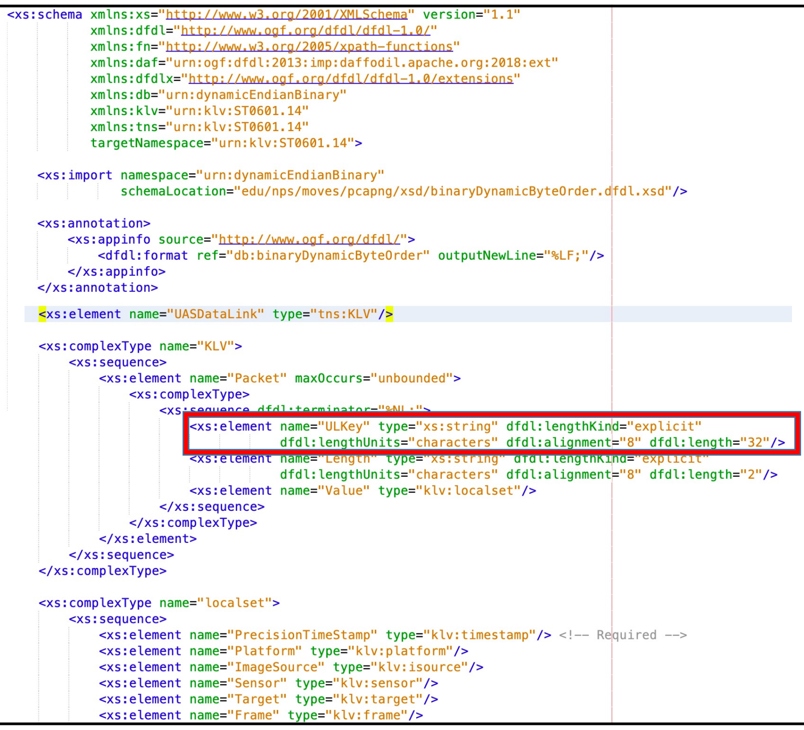
20

XSLT extraction of KLV String HEX

```
060E2B34020B01010E010301010000007802080005BC96253C937C05025C2A06020C4907021D250A095363616E4561676C650B0D4469676974616C20566
060E2B34020B01010E010301010000007802080005BC96253FA33A05025C480602091807021B970A095363616E4561676C650B0D4469676974616C20566
060E2B34020B01010E010301010000007802080005BC962542B33505025C8B0602059307021AB60A095363616E4561676C650B0D4469676974616C20566
060E2B34020B01010E010301010000007802080005BC962545C3AA05025D08060206AC07021B100A095363616E4561676C650B0D4469676974616C20566
060E2B34020B01010E010301010000007802080005BC962548D3C305025D53060207D907021BA60A095363616E4561676C650B0D4469676974616C20566
060E2B34020B01010E010301010000007802080005BC96254BE3A005025DC60602072607021C690A095363616E4561676C650B0D4469676974616C20566
060E2B34020B01010E010301010000007802080005BC96254EF3B905025E370602074307021C9E0A095363616E4561676C650B0D4469676974616C20566
060E2B34020B01010E010301010000007802080005BC962552035805025EBB0602088207021D8A0A095363616E4561676C650B0D4469676974616C20566
060E2B34020B01010E010301010000007802080005BC96255517BD05025F1F0602093407021E8D0A095363616E4561676C650B0D4469676974616C20566
060E2B34020B01010E010301010000007802080005BC962558277A05025FB606020BFE07021F020A095363616E4561676C650B0D4469676974616C20566
060E2B34020B01010E010301010000007802080005BC96255B373805025FF306020D2107021F1C0A095363616E4561676C650B0D4469676974616C20566
060E2B34020B01010E010301010000007802080005BC96255E46B9050260770602090F07021F6F0A095363616E4561676C650B0D4469676974616C20566
060E2B34020B01010E010301010000007802080005BC96256156D30502612106020A8607021F510A095363616E4561676C650B0D4469676974616C20566
060E2B34020B01010E010301010000007802080005BC9625646690050261B90602101907021EC20A095363616E4561676C650B0D4469676974616C20566
060E2B34020B01010E010301010000007802080005BC962567768B050261FD060211E507021ED80A095363616E4561676C650B0D4469676974616C20566
060E2B34020B01010E010301010000007802080005BC96256A862B05026266060210AF07021E190A095363616E4561676C650B0D4469676974616C20566
060E2B34020B01010E010301010000007802080005BC96256D958D050262B1060210FA07021D7F0A095363616E4561676C650B0D4469676974616C20566
060E2B34020B01010E010301010000007802080005BC962570A50E0502634A0602112907021CAD0A095363616E4561676C650B0D4469676974616C20566
060E2B34020B01010E010301010000007802080005BC962573B85F050263C4060214DD07021CE10A095363616E4561676C650B0D4469676974616C20566
060E2B34020B01010E010301010000007802080005BC962576C7E00502642A0602152807021CBC0A095363616E4561676C650B0D4469676974616C20566
060E2B34020B01010E010301010000007802080005BC962579D761050264BE060214E707021C870A095363616E4561676C650B0D4469676974616C20566
060E2B34020B01010E010301010000007802080005BC96257CE7F405026508060215BF07021BAA0A095363616E4561676C650B0D4469676974616C20566
060E2B34020B01010E010301010000007802080005BC96257FF8A6050265690602182107021A2E0A095363616E4561676C650B0D4469676974616C20566
060E2B34020B01010E010301010000007802080005BC9625830846050265A806021888070218DC0A095363616E4561676C650B0D4469676974616C20566
060E2B34020B01010E010301010000007802080005BC96258617A805026602060219E3070218BB0A095363616E4561676C650B0D4469676974616C20566
```

21

DFDL Decorated
XML Schema
for KLV



```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" version="1.1"
           xmlns:dfdl="http://www.ogf.org/dfdl/dfdl-1.0/"
           xmlns:fn="http://www.w3.org/2005/xpath-functions"
           xmlns:daf="urn:ogf:dfdl:2013:imp:daffodil.apache.org:2018:ext"
           xmlns:dfdlx="http://www.ogf.org/dfdl/dfdl-1.0/extensions"
           xmlns:db="urn:dynamicEndianBinary"
           xmlns:klv="urn:klv:ST0601.14"
           xmlns:tns="urn:klv:ST0601.14"
           targetNamespace="urn:klv:ST0601.14">

    <xs:import namespace="urn:dynamicEndianBinary"
               schemaLocation="edu/nps/moves/pcapng/xsd/binaryDynamicByteOrder.dfdl.xsd"/>

    <xs:annotation>
        <xs:appinfo source="http://www.ogf.org/dfdl/">
            <dfdl:format ref="db:binaryDynamicByteOrder" outputNewLine="%LF;"/>
        </xs:appinfo>
    </xs:annotation>

    <xs:element name="UASDataLink" type="tns:KLV"/>

    <xs:complexType name="KLV">
        <xs:sequence>
            <xs:element name="Packet" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence dfdl:terminator="%NL;">
                        <xs:element name="ULKey" type="xs:string" dfdl:lengthKind="explicit"
                                    dfdl:lengthUnits="characters" dfdl:alignment="8" dfdl:length="32"/>
                        <xs:element name="Length" type="xs:string" dfdl:lengthKind="explicit"
                                    dfdl:lengthUnits="characters" dfdl:alignment="8" dfdl:length="2"/>
                        <xs:element name="Value" type="klv:localset"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="localset">
        <xs:sequence>
            <xs:element name="PrecisionTimeStamp" type="klv:timestamp"/> <!-- Required -->
            <xs:element name="Platform" type="klv:platform"/>
            <xs:element name="ImageSource" type="klv:isource"/>
            <xs:element name="Sensor" type="klv:sensor"/>
            <xs:element name="Target" type="klv:target"/>
            <xs:element name="Frame" type="klv:frame"/>
```

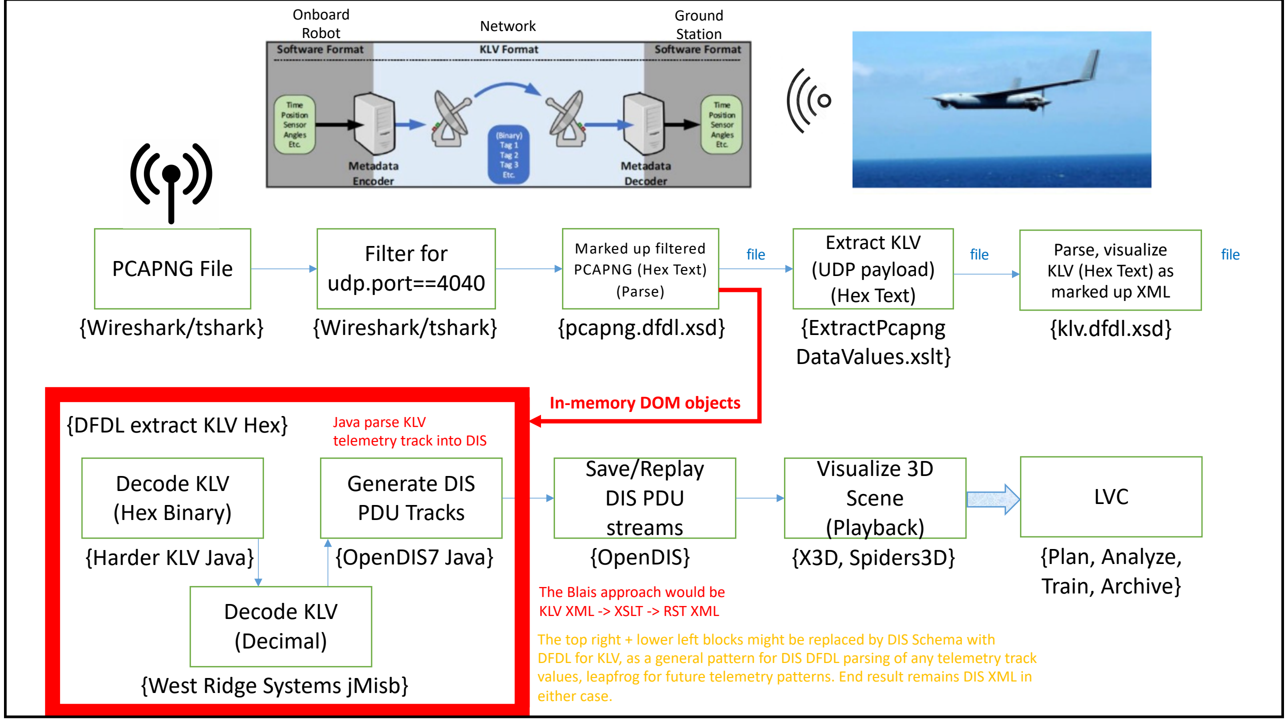
22

```xml
<?xml version="1.0" encoding="UTF-8"?>
<tns:UASDataLink xmlns:tns="urn:klv:ST0601.14">
  <Packet>
    <ULKey>060E2B34020B01010E01030101000000</ULKey>
    <Length>78</Length>
    <Value>
      <PrecisionTimeStamp>
        <Tag>02</Tag>
        <Length>08</Length>
        <Value>0005BC9691485FFB</Value>
      </PrecisionTimeStamp>
      <Platform>
        <HeadingAngle>
          <Tag>05</Tag>
          <Length>02</Length>
          <Value>214F</Value>
        </HeadingAngle>
        <PitchAngle>
          <Tag>06</Tag>
          <Length>02</Length>
          <Value>0FD7</Value>
        </PitchAngle>
        <RollAngle>
          <Tag>07</Tag>
          <Length>02</Length>
          <Value>19AB</Value>
        </RollAngle>
```

Parsed KLV in XML Format

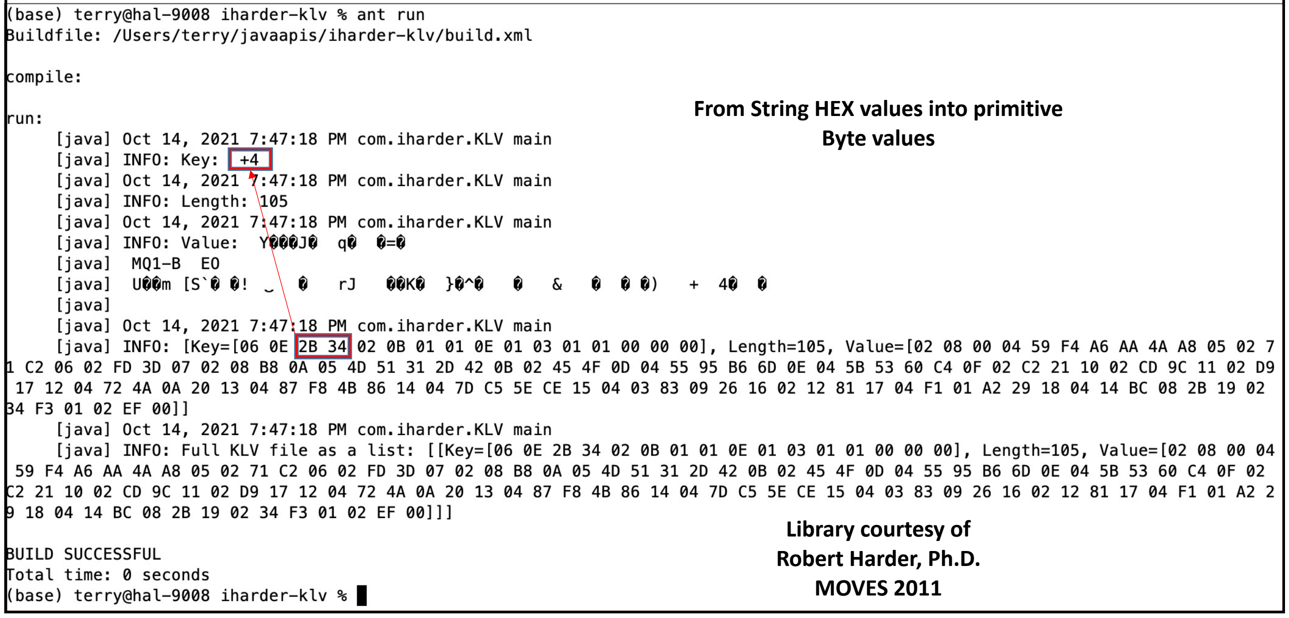
```xml
      <Sensor>
        <Latitude>
          <Tag>0D</Tag>
          <Length>04</Length>
          <Value>32DFF147</Value>
        </Latitude>
        <Longitude>
          <Tag>0E</Tag>
          <Length>04</Length>
          <Value>AA1C7BC3</Value>
        </Longitude>
        <TrueAltitude>
          <Tag>0F</Tag>
          <Length>02</Length>
          <Value>174A</Value>
        </TrueAltitude>
        <HorizontalFOV>
          <Tag>10</Tag>
          <Length>02</Length>
          <Value>4540</Value>
        </HorizontalFOV>
        <VerticalFOV>
          <Tag>11</Tag>
          <Length>02</Length>
          <Value>33F0</Value>
        </VerticalFOV>
```

23


Onboard Robot
Network
Ground Station
Software Format
KLV Format
Software Format
Time Position Sensor Angles Etc.
(Binary) Tag 1 Tag 2 Tag 3 Etc.
Metadata Encoder
Metadata Decoder
PCAPNG File
{Wireshark/tshark}
Filter for udp.port==4040
{Wireshark/tshark}
Marked up filtered PCAPNG (Hex Text) (Parse)
{pcapng.dfdl.xsd}
file
Extract KLV (UDP payload) (Hex Text)
{ExtractPcapng DataValues.xslt}
file
Parse, visualize KLV (Hex Text) as marked up XML
{klv.dfdl.xsd}
file
In-memory DOM objects
{DFDL extract KLV Hex}
Java parse KLV telemetry track into DIS
Decode KLV (Hex Binary)
{Harder KLV Java}
Generate DIS PDU Tracks
{OpenDIS7 Java}
Decode KLV (Decimal)
{West Ridge Systems jMisb}
Save/Replay DIS PDU streams
{OpenDIS}
Visualize 3D Scene (Playback)
{X3D, Spiders3D}
LVC
{Plan, Analyze, Train, Archive}
The Blais approach would be KLV XML -> XSLT -> RST XML
The top right + lower left blocks might be replaced by DIS Schema with DFDL for KLV, as a general pattern for DIS DFDL parsing of any telemetry track values, leapfrog for future telemetry patterns. End result remains DIS XML in either case.

24

Harder KLV



```
(base) terry@hal-9008 iharder-klv % ant run
Buildfile: /Users/terry/javaapis/iharder-klv/build.xml

compile:

run:
     [java] Oct 14, 2021 7:47:18 PM com.iharder.KLV main
     [java] INFO: Key:  +4
     [java] Oct 14, 2021 7:47:18 PM com.iharder.KLV main
     [java] INFO: Length: 105
     [java] Oct 14, 2021 7:47:18 PM com.iharder.KLV main
     [java] INFO: Value:  Y���J�  q�  �=�
     [java]  MQ1-B  EO
     [java]  U��m [S`� �! ‿  �   rJ   ��K�  }�^�   �   &   �  � �)   +  4�  �
     [java]
     [java] Oct 14, 2021 7:47:18 PM com.iharder.KLV main
     [java] INFO: [Key=[06 0E 2B 34 02 0B 01 01 0E 01 03 01 01 00 00 00], Length=105, Value=[02 08 00 04 59 F4 A6 AA 4A A8 05 02 7
1 C2 06 02 FD 3D 07 02 08 B8 0A 05 4D 51 31 2D 42 0B 02 45 4F 0D 04 55 95 B6 6D 0E 04 5B 53 60 C4 0F 02 C2 21 10 02 CD 9C 11 02 D9
 17 12 04 72 4A 0A 20 13 04 87 F8 4B 86 14 04 7D C5 5E CE 15 04 03 83 09 26 16 02 12 81 17 04 F1 01 A2 29 18 04 14 BC 08 2B 19 02
34 F3 01 02 EF 00]]
     [java] Oct 14, 2021 7:47:18 PM com.iharder.KLV main
     [java] INFO: Full KLV file as a list: [[Key=[06 0E 2B 34 02 0B 01 01 0E 01 03 01 01 00 00 00], Length=105, Value=[02 08 00 04
 59 F4 A6 AA 4A A8 05 02 71 C2 06 02 FD 3D 07 02 08 B8 0A 05 4D 51 31 2D 42 0B 02 45 4F 0D 04 55 95 B6 6D 0E 04 5B 53 60 C4 0F 02
C2 21 10 02 CD 9C 11 02 D9 17 12 04 72 4A 0A 20 13 04 87 F8 4B 86 14 04 7D C5 5E CE 15 04 03 83 09 26 16 02 12 81 17 04 F1 01 A2 2
9 18 04 14 BC 08 2B 19 02 34 F3 01 02 EF 00]]]

BUILD SUCCESSFUL
Total time: 0 seconds
(base) terry@hal-9008 iharder-klv %
```

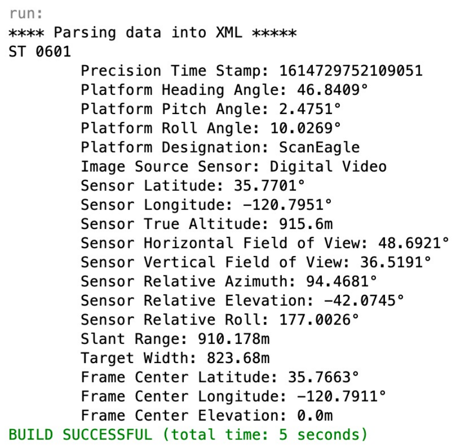
From String HEX values into primitive Byte values

Library courtesy of Robert Harder, Ph.D. MOVES 2011

25

West Ridge Systems jMisb

KLV Decoded decimal values

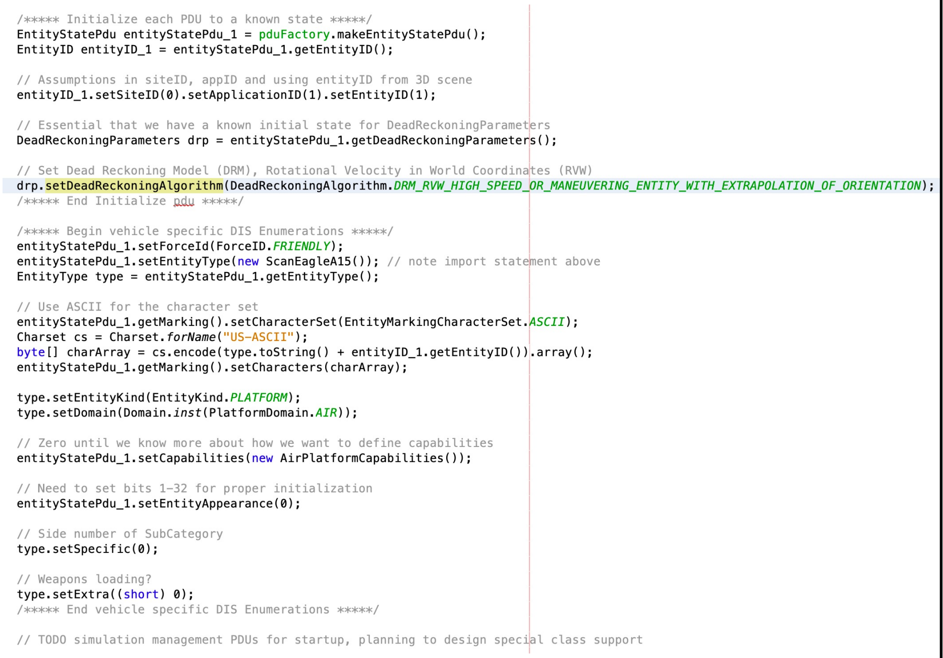


```
run:
**** Parsing data into XML *****
ST 0601
        Precision Time Stamp: 1614729752109051
        Platform Heading Angle: 46.8409°
        Platform Pitch Angle: 2.4751°
        Platform Roll Angle: 10.0269°
        Platform Designation: ScanEagle
        Image Source Sensor: Digital Video
        Sensor Latitude: 35.7701°
        Sensor Longitude: -120.7951°
        Sensor True Altitude: 915.6m
        Sensor Horizontal Field of View: 48.6921°
        Sensor Vertical Field of View: 36.5191°
        Sensor Relative Azimuth: 94.4681°
        Sensor Relative Elevation: -42.0745°
        Sensor Relative Roll: 177.0026°
        Slant Range: 910.178m
        Target Width: 823.68m
        Frame Center Latitude: 35.7663°
        Frame Center Longitude: -120.7911°
        Frame Center Elevation: 0.0m
BUILD SUCCESSFUL (total time: 5 seconds)
```

26

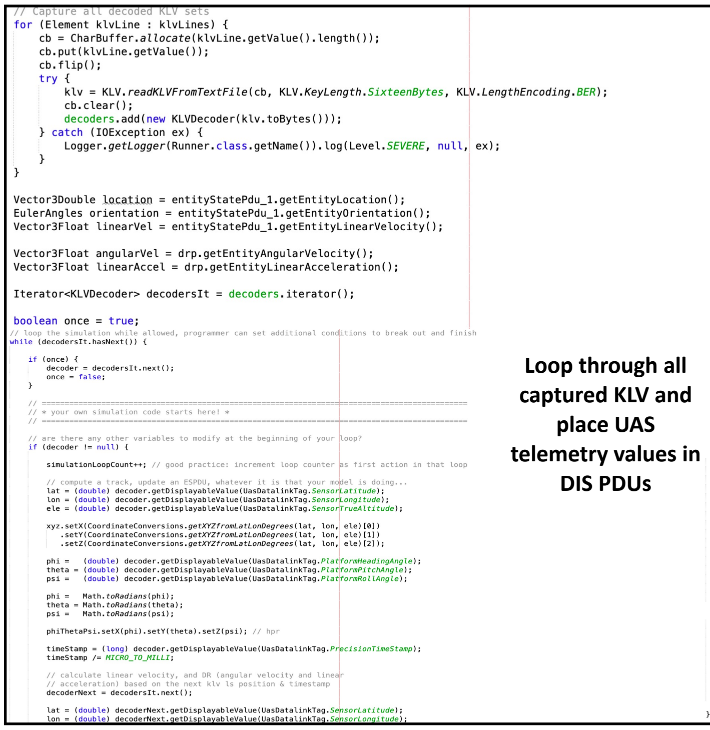
Set up the DIS EntityStatePDU

OpenDIS7 Java



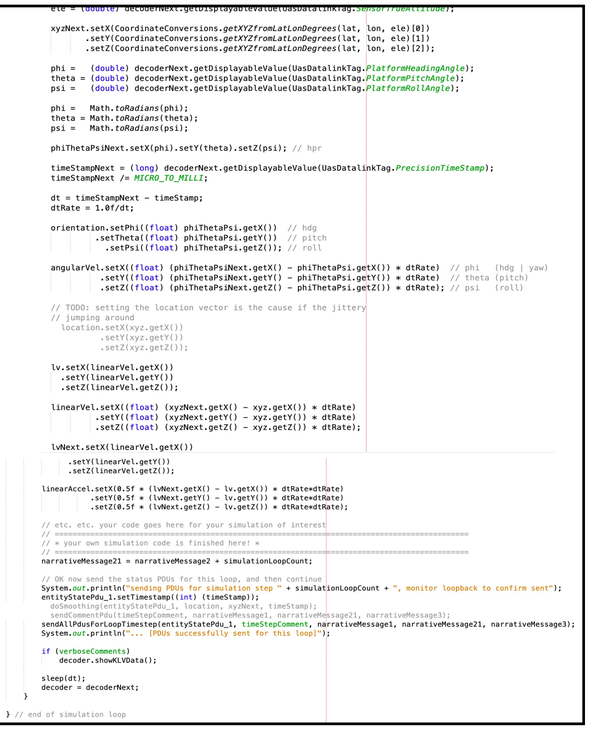
```
/***** Initialize each PDU to a known state *****/
EntityStatePdu entityStatePdu_1 = pduFactory.makeEntityStatePdu();
EntityID entityID_1 = entityStatePdu_1.getEntityID();

// Assumptions in siteID, appID and using entityID from 3D scene
entityID_1.setSiteID(0).setApplicationID(1).setEntityID(1);

// Essential that we have a known initial state for DeadReckoningParameters
DeadReckoningParameters drp = entityStatePdu_1.getDeadReckoningParameters();

// Set Dead Reckoning Model (DRM), Rotational Velocity in World Coordinates (RVW)
drp.setDeadReckoningAlgorithm(DeadReckoningAlgorithm.DRM_RVW_HIGH_SPEED_OR_MANEUVERING_ENTITY_WITH_EXTRAPOLATION_OF_ORIENTATION);
/***** End Initialize pdu *****/

/***** Begin vehicle specific DIS Enumerations *****/
entityStatePdu_1.setForceId(ForceID.FRIENDLY);
entityStatePdu_1.setEntityType(new ScanEagleA15()); // note import statement above
EntityType type = entityStatePdu_1.getEntityType();

// Use ASCII for the character set
entityStatePdu_1.getMarking().setCharacterSet(EntityMarkingCharacterSet.ASCII);
Charset cs = Charset.forName("US-ASCII");
byte[] charArray = cs.encode(type.toString() + entityID_1.getEntityID()).array();
entityStatePdu_1.getMarking().setCharacters(charArray);

type.setEntityKind(EntityKind.PLATFORM);
type.setDomain(Domain.inst(PlatformDomain.AIR));

// Zero until we know more about how we want to define capabilities
entityStatePdu_1.setCapabilities(new AirPlatformCapabilities());

// Need to set bits 1-32 for proper initialization
entityStatePdu_1.setEntityAppearance(0);

// Side number of SubCategory
type.setSpecific(0);

// Weapons loading?
type.setExtra((short) 0);
/***** End vehicle specific DIS Enumerations *****/

// TODO simulation management PDUs for startup, planning to design special class support
```

27



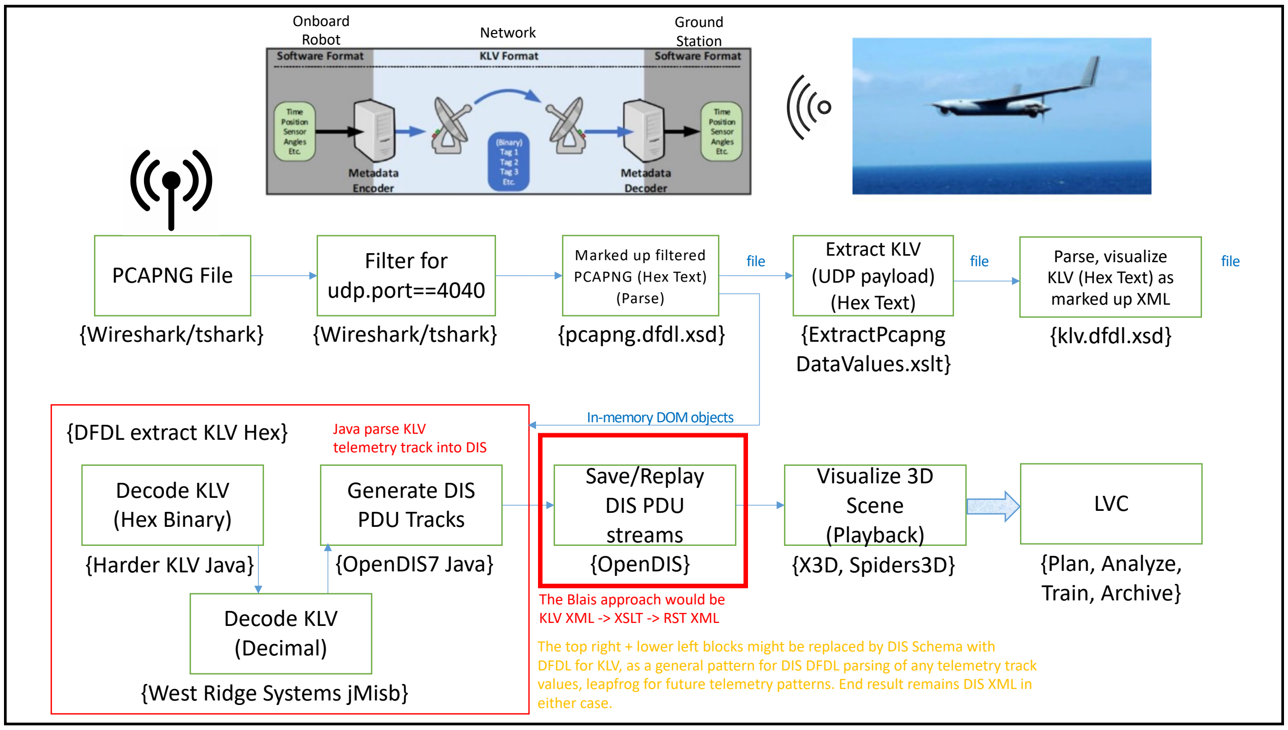
```
// Capture all decoded KLV sets
for (Element klvLine : klvLines) {
    cb = CharBuffer.allocate(klvLine.getValue().length());
    cb.put(klvLine.getValue());
    cb.flip();
    try {
        klv = KLV.readKLVFromTextFile(cb, KLV.KeyLength.SixteenBytes, KLV.LengthEncoding.BER);
        cb.clear();
        decoders.add(new KLVDecoder(klv.toBytes()));
    } catch (IOException ex) {
        Logger.getLogger(Runner.class.getName()).log(Level.SEVERE, null, ex);
    }
}

Vector3Double location = entityStatePdu_1.getEntityLocation();
EulerAngles orientation = entityStatePdu_1.getEntityOrientation();
Vector3Float linearVel = entityStatePdu_1.getEntityLinearVelocity();

Vector3Float angularVel = drp.getEntityAngularVelocity();
Vector3Float linearAccel = drp.getEntityLinearAcceleration();

Iterator<KLVDecoder> decodersIt = decoders.iterator();

boolean once = true;
// loop the simulation while allowed, programmer can set additional conditions to break out and finish
while (decodersIt.hasNext()) {

    if (once) {
        decoder = decodersIt.next();
        once = false;
    }

    // ==========================================================================================
    // * your own simulation code starts here! *
    // ==========================================================================================

    // are there any other variables to modify at the beginning of your loop?
    if (decoder != null) {

        simulationLoopCount++; // good practice: increment loop counter as first action in that loop

        // compute a track, update an ESPDU, whatever it is that your model is doing...
        lat = (double) decoder.getDisplayableValue(UasDatalinkTag.SensorLatitude);
        lon = (double) decoder.getDisplayableValue(UasDatalinkTag.SensorLongitude);
        ele = (double) decoder.getDisplayableValue(UasDatalinkTag.SensorTrueAltitude);

        xyz.setX(CoordinateConversions.getXYZfromLatLonDegrees(lat, lon, ele)[0])
           .setY(CoordinateConversions.getXYZfromLatLonDegrees(lat, lon, ele)[1])
           .setZ(CoordinateConversions.getXYZfromLatLonDegrees(lat, lon, ele)[2]);

        phi =   (double) decoder.getDisplayableValue(UasDatalinkTag.PlatformHeadingAngle);
        theta = (double) decoder.getDisplayableValue(UasDatalinkTag.PlatformPitchAngle);
        psi =   (double) decoder.getDisplayableValue(UasDatalinkTag.PlatformRollAngle);

        phi =   Math.toRadians(phi);
        theta = Math.toRadians(theta);
        psi =   Math.toRadians(psi);

        phiThetaPsi.setX(phi).setY(theta).setZ(psi); // hpr

        timeStamp = (long) decoder.getDisplayableValue(UasDatalinkTag.PrecisionTimeStamp);
        timeStamp /= MICRO_TO_MILLI;

        // calculate linear velocity, and DR (angular velocity and linear
        // acceleration) based on the next klv ls position & timestamp
        decoderNext = decodersIt.next();

        lat = (double) decoderNext.getDisplayableValue(UasDatalinkTag.SensorLatitude);
        lon = (double) decoderNext.getDisplayableValue(UasDatalinkTag.SensorLongitude);
```

Loop through all captured KLV and place UAS telemetry values in DIS PDUs



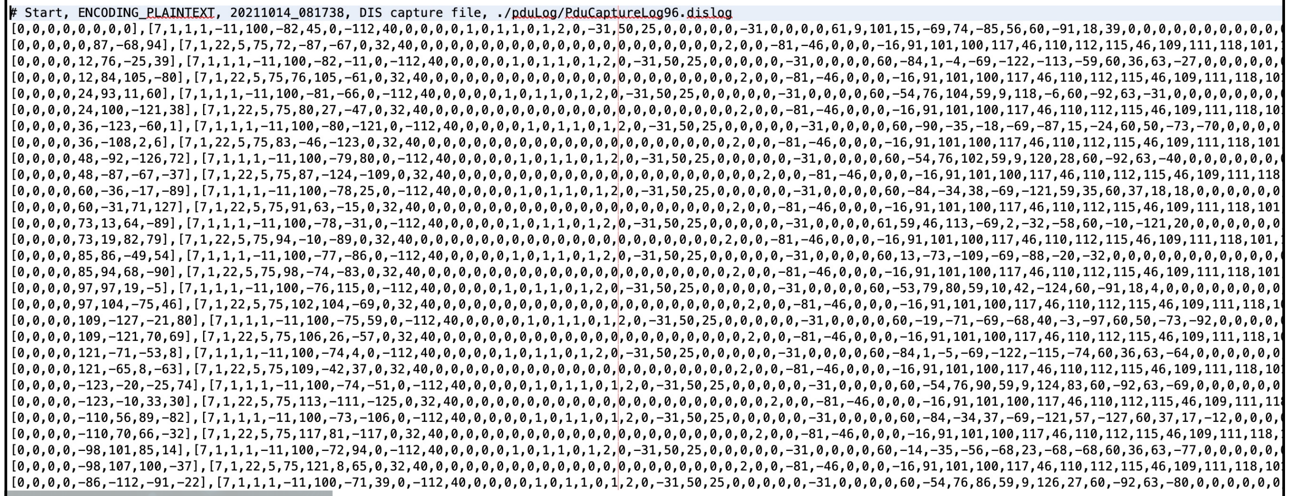
```
        ele = (double) decoderNext.getDisplayableValue(UasDatalinkTag.SensorTrueAltitude);

        xyzNext.setX(CoordinateConversions.getXYZfromLatLonDegrees(lat, lon, ele)[0])
               .setY(CoordinateConversions.getXYZfromLatLonDegrees(lat, lon, ele)[1])
               .setZ(CoordinateConversions.getXYZfromLatLonDegrees(lat, lon, ele)[2]);

        phi =   (double) decoderNext.getDisplayableValue(UasDatalinkTag.PlatformHeadingAngle);
        theta = (double) decoderNext.getDisplayableValue(UasDatalinkTag.PlatformPitchAngle);
        psi =   (double) decoderNext.getDisplayableValue(UasDatalinkTag.PlatformRollAngle);

        phi =   Math.toRadians(phi);
        theta = Math.toRadians(theta);
        psi =   Math.toRadians(psi);

        phiThetaPsiNext.setX(phi).setY(theta).setZ(psi); // hpr

        timeStampNext = (long) decoderNext.getDisplayableValue(UasDatalinkTag.PrecisionTimeStamp);
        timeStampNext /= MICRO_TO_MILLI;

        dt = timeStampNext - timeStamp;
        dtRate = 1.0f/dt;

        orientation.setPhi((float) phiThetaPsi.getX())  // hdg
                .setTheta((float) phiThetaPsi.getY())  // pitch
                  .setPsi((float) phiThetaPsi.getZ()); // roll

        angularVel.setX((float) (phiThetaPsiNext.getX() - phiThetaPsi.getX()) * dtRate)  // phi   (hdg | yaw)
                  .setY((float) (phiThetaPsiNext.getY() - phiThetaPsi.getY()) * dtRate)  // theta (pitch)
                  .setZ((float) (phiThetaPsiNext.getZ() - phiThetaPsi.getZ()) * dtRate); // psi   (roll)

        // TODO: setting the location vector is the cause if the jittery
        // jumping around
          location.setX(xyz.getX())
                  .setY(xyz.getY())
                  .setZ(xyz.getZ());

        lv.setX(linearVel.getX())
          .setY(linearVel.getY())
          .setZ(linearVel.getZ());

        linearVel.setX((float) (xyzNext.getX() - xyz.getX()) * dtRate)
                 .setY((float) (xyzNext.getY() - xyz.getY()) * dtRate)
                 .setZ((float) (xyzNext.getZ() - xyz.getZ()) * dtRate);

        lvNext.setX(linearVel.getX())
            .setY(linearVel.getY())
            .setZ(linearVel.getZ());

        linearAccel.setX(0.5f * (lvNext.getX() - lv.getX()) * dtRate*dtRate)
                   .setY(0.5f * (lvNext.getY() - lv.getY()) * dtRate*dtRate)
                   .setZ(0.5f * (lvNext.getZ() - lv.getZ()) * dtRate*dtRate);

        // etc. etc. your code goes here for your simulation of interest
        // ==========================================================================================
        // * your own simulation code is finished here! *
        // ==========================================================================================
        narrativeMessage21 = narrativeMessage2 + simulationLoopCount;

        // OK now send the status PDUs for this loop, and then continue
        System.out.println("sending PDUs for simulation step " + simulationLoopCount + ", monitor loopback to confirm sent");
        entityStatePdu_1.setTimestamp((int) (timeStamp));
          doSmoothing(entityStatePdu_1, location, xyzNext, timeStamp);
          sendCommentPdu(timeStepComment, narrativeMessage1, narrativeMessage21, narrativeMessage3);
        sendAllPdusForLoopTimestep(entityStatePdu_1, timeStepComment, narrativeMessage1, narrativeMessage21, narrativeMessage3);
        System.out.println("... [PDUs successfully sent for this loop]");

        if (verboseComments)
            decoder.showKLVData();

        sleep(dt);
        decoder = decoderNext;
    }

} // end of simulation loop
```

28

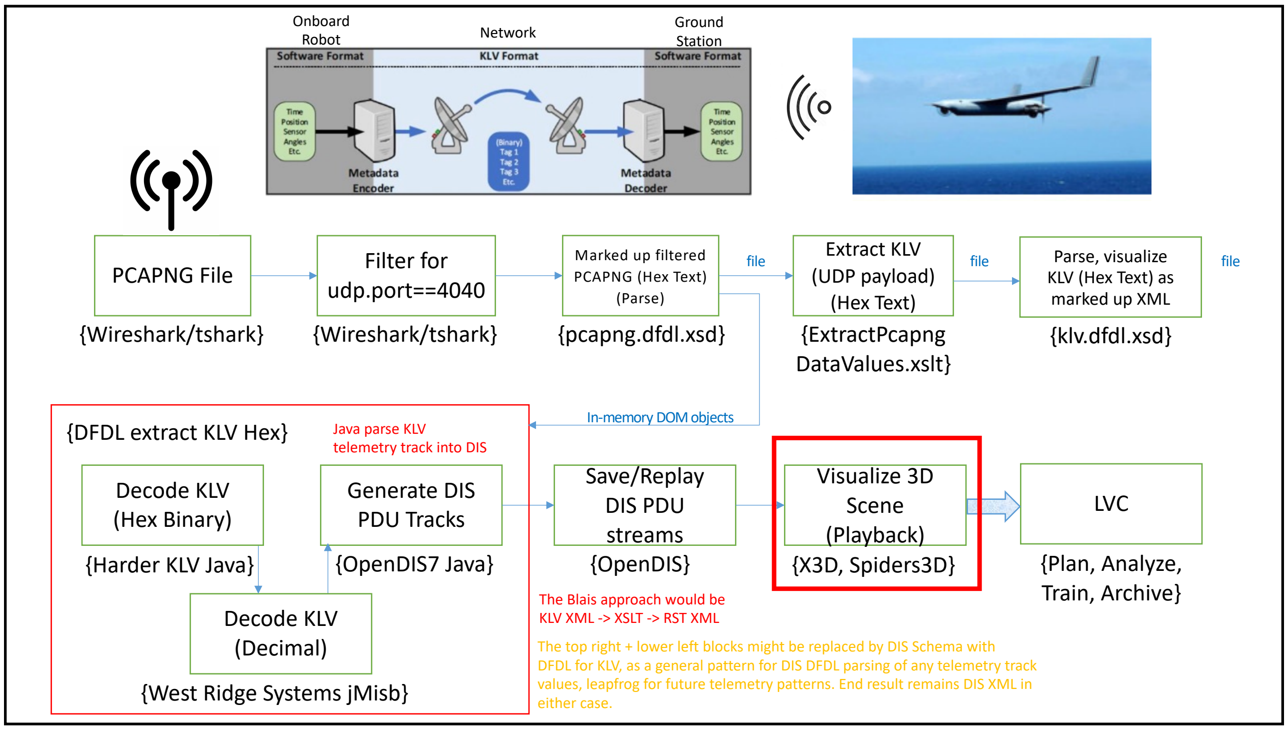

Onboard Robot
Network
Ground Station
Software Format
KLV Format
Software Format
Time Position Sensor Angles Etc.
Metadata Encoder
(Binary) Tag 1 Tag 2 Tag 3 Etc.
Metadata Decoder
Time Position Sensor Angles Etc.
PCAPNG File
{Wireshark/tshark}
Filter for udp.port==4040
{Wireshark/tshark}
Marked up filtered PCAPNG (Hex Text) (Parse)
{pcapng.dfdl.xsd}
file
Extract KLV (UDP payload) (Hex Text)
{ExtractPcapng DataValues.xslt}
file
Parse, visualize KLV (Hex Text) as marked up XML
{klv.dfdl.xsd}
file
In-memory DOM objects
{DFDL extract KLV Hex}
Java parse KLV telemetry track into DIS
Decode KLV (Hex Binary)
{Harder KLV Java}
Generate DIS PDU Tracks
{OpenDIS7 Java}
Decode KLV (Decimal)
{West Ridge Systems jMisb}
Save/Replay DIS PDU streams
{OpenDIS}
The Blais approach would be KLV XML -> XSLT -> RST XML
Visualize 3D Scene (Playback)
{X3D, Spiders3D}
LVC
{Plan, Analyze, Train, Archive}
The top right + lower left blocks might be replaced by DIS Schema with DFDL for KLV, as a general pattern for DIS DFDL parsing of any telemetry track values, leapfrog for future telemetry patterns. End result remains DIS XML in either case.

29

Saved DIS PDU Capture

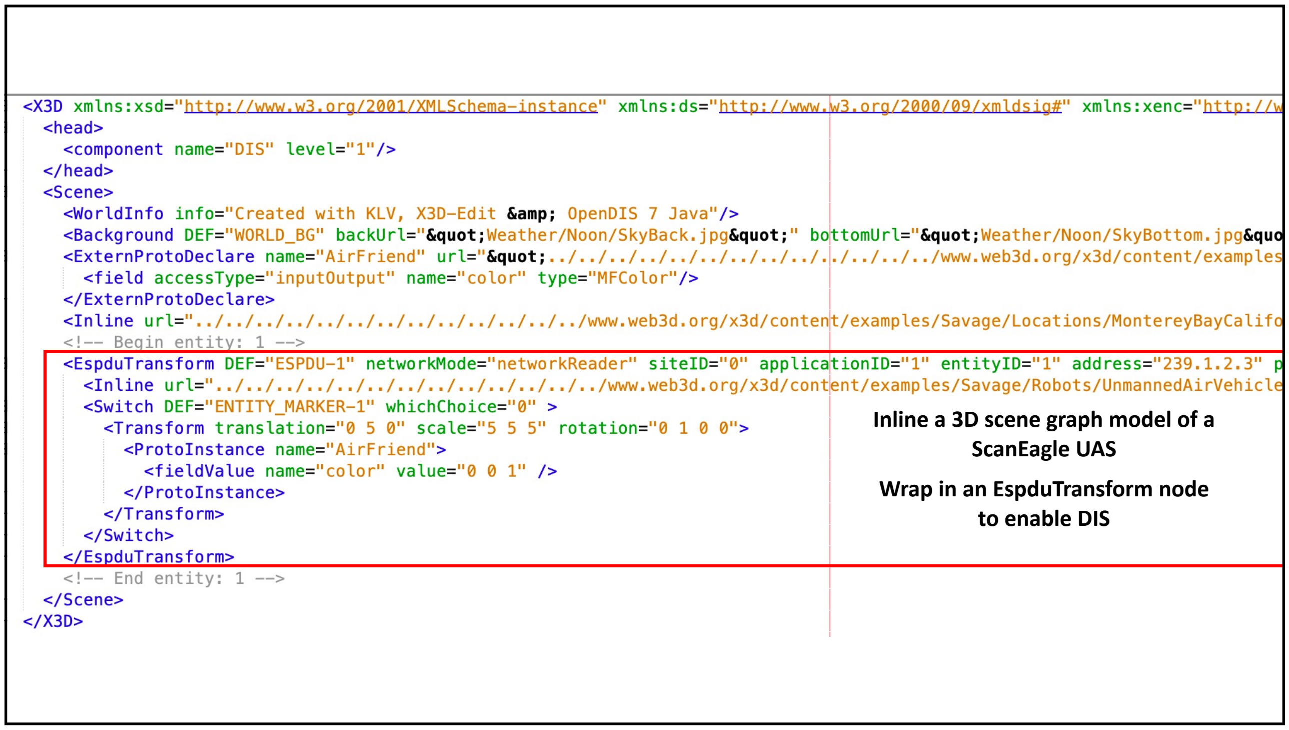

```
# Start, ENCODING_PLAINTEXT, 20211014_081738, DIS capture file, ./pduLog/PduCaptureLog96.dislog
[0,0,0,0,0,0,0,0],[7,1,1,1,-11,100,-82,45,0,-112,40,0,0,0,0,1,0,1,1,0,1,2,0,-31,50,25,0,0,0,0,0,-31,0,0,0,0,61,9,101,15,-69,74,-85,56,60,-91,18,39,0,0,0,0,0,0,0,0,0,0,
[0,0,0,0,0,87,-68,94],[7,1,22,5,75,72,-87,-67,0,32,40,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,-81,-46,0,0,0,-16,91,101,100,117,46,110,112,115,46,109,111,118,101,
[0,0,0,0,12,76,-25,39],[7,1,1,1,-11,100,-82,-11,0,-112,40,0,0,0,0,1,0,1,1,0,1,2,0,-31,50,25,0,0,0,0,0,-31,0,0,0,0,60,-84,1,-4,-69,-122,-113,-59,60,36,63,-27,0,0,0,0,0,
[0,0,0,0,12,84,105,-80],[7,1,22,5,75,76,105,-61,0,32,40,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,-81,-46,0,0,0,-16,91,101,100,117,46,110,112,115,46,109,111,118,10
[0,0,0,0,24,93,11,60],[7,1,1,1,-11,100,-81,-66,0,-112,40,0,0,0,0,1,0,1,1,0,1,2,0,-31,50,25,0,0,0,0,0,-31,0,0,0,0,60,-54,76,104,59,9,118,-6,60,-92,63,-31,0,0,0,0,0,0,
[0,0,0,0,24,100,-121,38],[7,1,22,5,75,80,27,-47,0,32,40,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,-81,-46,0,0,0,-16,91,101,100,117,46,110,112,115,46,109,111,118,10
[0,0,0,0,36,-123,-60,1],[7,1,1,1,-11,100,-80,-121,0,-112,40,0,0,0,0,1,0,1,1,0,1,2,0,-31,50,25,0,0,0,0,0,-31,0,0,0,0,60,-90,-35,-18,-69,-87,15,-24,60,50,-73,-70,0,0,0,0,0
[0,0,0,0,36,-108,2,6],[7,1,22,5,75,83,-46,-123,0,32,40,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,-81,-46,0,0,0,-16,91,101,100,117,46,110,112,115,46,109,111,118,101
[0,0,0,0,48,-92,-126,72],[7,1,1,1,-11,100,-79,80,0,-112,40,0,0,0,0,1,0,1,1,0,1,2,0,-31,50,25,0,0,0,0,0,-31,0,0,0,0,60,-54,76,102,59,9,120,28,60,-92,63,-40,0,0,0,0,0,0,
[0,0,0,0,48,-87,-67,-37],[7,1,22,5,75,87,-124,-109,0,32,40,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,-81,-46,0,0,0,-16,91,101,100,117,46,110,112,115,46,109,111,118
[0,0,0,0,60,-36,-17,-89],[7,1,1,1,-11,100,-78,25,0,-112,40,0,0,0,0,1,0,1,1,0,1,2,0,-31,50,25,0,0,0,0,0,-31,0,0,0,0,60,-84,-34,38,-69,-121,59,35,60,37,18,18,0,0,0,0,0,0
[0,0,0,0,60,-31,71,127],[7,1,22,5,75,91,63,-15,0,32,40,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,-81,-46,0,0,0,-16,91,101,100,117,46,110,112,115,46,109,111,118,101
[0,0,0,0,73,13,64,-89],[7,1,1,1,-11,100,-78,-31,0,-112,40,0,0,0,0,1,0,1,1,0,1,2,0,-31,50,25,0,0,0,0,0,-31,0,0,0,0,61,59,46,113,-69,2,-32,-58,60,-10,-121,20,0,0,0,0,0,0
[0,0,0,0,73,19,82,79],[7,1,22,5,75,94,-10,-89,0,32,40,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,-81,-46,0,0,0,-16,91,101,100,117,46,110,112,115,46,109,111,118,101,
[0,0,0,0,85,86,-49,54],[7,1,1,1,-11,100,-77,-86,0,-112,40,0,0,0,0,1,0,1,1,0,1,2,0,-31,50,25,0,0,0,0,0,-31,0,0,0,0,60,13,-73,-109,-69,-88,-20,-32,0,0,0,0,0,0,0,0,0,0,
[0,0,0,0,85,94,68,-90],[7,1,22,5,75,98,-74,-83,0,32,40,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,-81,-46,0,0,0,-16,91,101,100,117,46,110,112,115,46,109,111,118,101
[0,0,0,0,97,97,19,-5],[7,1,1,1,-11,100,-76,115,0,-112,40,0,0,0,0,1,0,1,1,0,1,2,0,-31,50,25,0,0,0,0,0,-31,0,0,0,0,60,-53,79,80,59,10,42,-124,60,-91,18,4,0,0,0,0,0,0,0,0
[0,0,0,0,97,104,-75,46],[7,1,22,5,75,102,104,-69,0,32,40,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,-81,-46,0,0,0,-16,91,101,100,117,46,110,112,115,46,109,111,118,1
[0,0,0,0,109,-127,-21,80],[7,1,1,1,-11,100,-75,59,0,-112,40,0,0,0,0,1,0,1,1,0,1,2,0,-31,50,25,0,0,0,0,0,-31,0,0,0,0,60,-19,-71,-69,-68,40,-3,-97,60,50,-73,-92,0,0,0,0,
[0,0,0,0,109,-121,70,69],[7,1,22,5,75,106,26,-57,0,32,40,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,-81,-46,0,0,0,-16,91,101,100,117,46,110,112,115,46,109,111,118,1
[0,0,0,0,121,-71,-53,8],[7,1,1,1,-11,100,-74,4,0,-112,40,0,0,0,0,1,0,1,1,0,1,2,0,-31,50,25,0,0,0,0,0,-31,0,0,0,0,60,-84,1,-5,-69,-122,-115,-74,60,36,63,-64,0,0,0,0,0,0
[0,0,0,0,121,-65,8,-63],[7,1,22,5,75,109,-42,37,0,32,40,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,-81,-46,0,0,0,-16,91,101,100,117,46,110,112,115,46,109,111,118,10
[0,0,0,0,-123,-20,-25,74],[7,1,1,1,-11,100,-74,-51,0,-112,40,0,0,0,0,1,0,1,1,0,1,2,0,-31,50,25,0,0,0,0,0,-31,0,0,0,0,60,-54,76,90,59,9,124,83,60,-92,63,-69,0,0,0,0,0,0
[0,0,0,0,-123,-10,33,30],[7,1,22,5,75,113,-111,-125,0,32,40,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,-81,-46,0,0,0,-16,91,101,100,117,46,110,112,115,46,109,111,11
[0,0,0,0,-110,56,89,-82],[7,1,1,1,-11,100,-73,-106,0,-112,40,0,0,0,0,1,0,1,1,0,1,2,0,-31,50,25,0,0,0,0,0,-31,0,0,0,0,60,-84,-34,37,-69,-121,57,-127,60,37,17,-12,0,0,0,
[0,0,0,0,-110,70,66,-32],[7,1,22,5,75,117,81,-117,0,32,40,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,-81,-46,0,0,0,-16,91,101,100,117,46,110,112,115,46,109,111,118,
[0,0,0,0,-98,101,85,14],[7,1,1,1,-11,100,-72,94,0,-112,40,0,0,0,0,1,0,1,1,0,1,2,0,-31,50,25,0,0,0,0,0,-31,0,0,0,0,60,-14,-35,-56,-68,23,-68,-68,60,36,63,-77,0,0,0,0,0,
[0,0,0,0,-98,107,100,-37],[7,1,22,5,75,121,8,65,0,32,40,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,-81,-46,0,0,0,-16,91,101,100,117,46,110,112,115,46,109,111,118,10
[0,0,0,0,-86,-112,-91,-22],[7,1,1,1,-11,100,-71,39,0,-112,40,0,0,0,0,1,0,1,1,0,1,2,0,-31,50,25,0,0,0,0,0,-31,0,0,0,0,60,-54,76,86,59,9,126,27,60,-92,63,-80,0,0,0,0,0,0
```

30



Onboard Robot
Network
Ground Station
Software Format
KLV Format
Software Format
Time Position Sensor Angles Etc.
Metadata Encoder
(Binary) Tag 1 Tag 2 Tag 3 Etc.
Metadata Decoder
Time Position Sensor Angles Etc.
PCAPNG File
{Wireshark/tshark}
Filter for udp.port==4040
{Wireshark/tshark}
Marked up filtered PCAPNG (Hex Text) (Parse)
{pcapng.dfdl.xsd}
file
Extract KLV (UDP payload) (Hex Text)
{ExtractPcapng DataValues.xslt}
file
Parse, visualize KLV (Hex Text) as marked up XML
{klv.dfdl.xsd}
file
In-memory DOM objects
{DFDL extract KLV Hex}
Java parse KLV telemetry track into DIS
Decode KLV (Hex Binary)
{Harder KLV Java}
Generate DIS PDU Tracks
{OpenDIS7 Java}
Decode KLV (Decimal)
{West Ridge Systems jMisb}
Save/Replay DIS PDU streams
{OpenDIS}
Visualize 3D Scene (Playback)
{X3D, Spiders3D}
LVC
{Plan, Analyze, Train, Archive}
The Blais approach would be KLV XML -> XSLT -> RST XML
The top right + lower left blocks might be replaced by DIS Schema with DFDL for KLV, as a general pattern for DIS DFDL parsing of any telemetry track values, leapfrog for future telemetry patterns. End result remains DIS XML in either case.

31

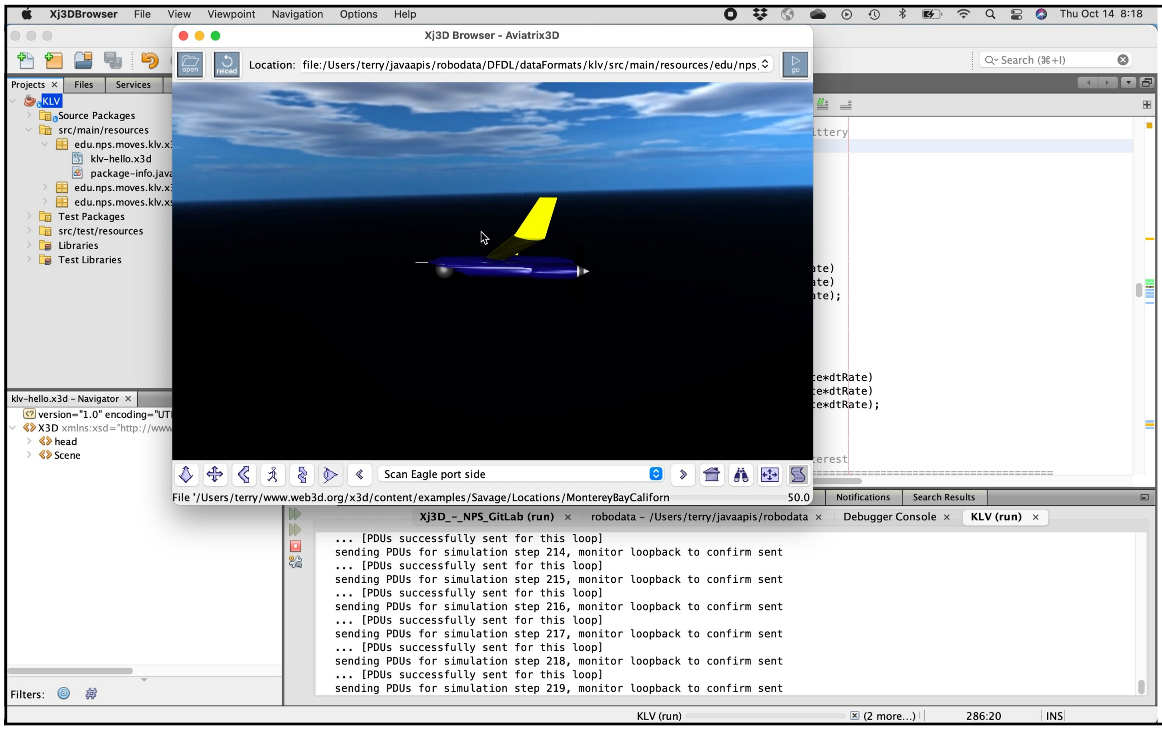


```
<X3D xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance" xmlns:ds="http://www.w3.org/2000/09/xmldsig#" xmlns:xenc="http://w
  <head>
    <component name="DIS" level="1"/>
  </head>
  <Scene>
    <WorldInfo info="Created with KLV, X3D-Edit &amp; OpenDIS 7 Java"/>
    <Background DEF="WORLD_BG" backUrl="&quot;Weather/Noon/SkyBack.jpg&quot;" bottomUrl="&quot;Weather/Noon/SkyBottom.jpg&quo
    <ExternProtoDeclare name="AirFriend" url="&quot;../../../../../../../../../../../../../www.web3d.org/x3d/content/examples
      <field accessType="inputOutput" name="color" type="MFColor"/>
    </ExternProtoDeclare>
    <Inline url="../../../../../../../../../../../../../www.web3d.org/x3d/content/examples/Savage/Locations/MontereyBayCalifo
    <!-- Begin entity: 1 -->
    <EspduTransform DEF="ESPDU-1" networkMode="networkReader" siteID="0" applicationID="1" entityID="1" address="239.1.2.3" p
      <Inline url="../../../../../../../../../../../../../www.web3d.org/x3d/content/examples/Savage/Robots/UnmannedAirVehicle
      <Switch DEF="ENTITY_MARKER-1" whichChoice="0" >
        <Transform translation="0 5 0" scale="5 5 5" rotation="0 1 0 0">
          <ProtoInstance name="AirFriend">
            <fieldValue name="color" value="0 0 1" />
          </ProtoInstance>
        </Transform>
      </Switch>
    </EspduTransform>
    <!-- End entity: 1 -->
  </Scene>
</X3D>
```

Inline a 3D scene graph model of a ScanEagle UAS

Wrap in an EspduTransform node to enable DIS

32


Xj3DBrowser File View Viewpoint Navigation Options Help
Thu Oct 14 8:18
Xj3D Browser - Aviatrix3D
Location: file:/Users/terry/javaapis/robodata/DFDL/dataFormats/klv/src/main/resources/edu/nps
Projects Files Services
KLV
Source Packages
src/main/resources
edu.nps.moves.klv.x3
klv-hello.x3d
package-info.java
Test Packages
src/test/resources
Libraries
Test Libraries
klv-hello.x3d - Navigator
head
Scene
Scan Eagle port side
File '/Users/terry/www.web3d.org/x3d/content/examples/Savage/Locations/MontereyBayCaliforn
Xj3D_-_NPS_GitLab (run)
robodata - /Users/terry/javaapis/robodata
Debugger Console
KLV (run)
Notifications
Search Results
Filters:
KLV (run)
(2 more...)
286:20
INS

```
... [PDUs successfully sent for this loop]
sending PDUs for simulation step 214, monitor loopback to confirm sent
... [PDUs successfully sent for this loop]
sending PDUs for simulation step 215, monitor loopback to confirm sent
... [PDUs successfully sent for this loop]
sending PDUs for simulation step 216, monitor loopback to confirm sent
... [PDUs successfully sent for this loop]
sending PDUs for simulation step 217, monitor loopback to confirm sent
... [PDUs successfully sent for this loop]
sending PDUs for simulation step 218, monitor loopback to confirm sent
... [PDUs successfully sent for this loop]
sending PDUs for simulation step 219, monitor loopback to confirm sent
```

I
t

w
o
r
k
s

☺

33

Time for In Memory PCAPNG Parse of 2,019 Lines of KLV to DIS Packet Generation

```
run:
[edu.nps.moves.klv.sim.Runner]started...
**** Parsing data into XML *****
[DisThreadedNetworkInterface] using network interface en0
[DisThreadedNetworkInterface] datagramSocket.joinGroup  address=239.1.2.3 port=3000 start() complete
Network confirmation: address=239.1.2.3 port=3000
Beginning pdu save to directory ./pduLog
Recorder log file open: /Users/terry/javaapis/robodata/DFDL/dataFormats/klv/pduLog/PduCaptureLog.dislog
[DisThreadedNetworkInterface] using network interface en0
[DisThreadedNetworkInterface] datagramSocket.joinGroup  address=239.1.2.3 port=3000 start() complete
[PduRecorder [edu.nps.moves.klv.sim.Runner] pduRecorder] listening to IP address 239.1.2.3 on port 3000
sending PDUs for simulation step 1, monitor loopback to confirm sent
... [PDUs successfully sent for this loop]
sending PDUs for simulation step 2, monitor loopback to confirm sent
... [PDUs successfully sent for this loop]
sending PDUs for simulation step 3, monitor loopback to confirm sent
... [PDUs successfully sent for this loop]
sending PDUs for simulation step 4, monitor loopback to confirm sent
... [PDUs successfully sent for this loop]
sending PDUs for simulation step 5, monitor loopback to confirm sent
... [PDUs successfully sent for this loop]
sending PDUs for simulation step 6, monitor loopback to confirm sent
... [PDUs successfully sent for this loop]
BUILD STOPPED (total time: 7 seconds)
```

34

Open-Source, Open-Standards Technology Used

- Wireshark
- PCAPNG
- ST 0601.17
- SMPTE ST 336
- Apache Daffodil
- OpenDIS7 Java
- Harder KLV
- West Ridge Systems jMisb
- X3D Graphics
- Xj3D
- XML (Schema, XSLT)
- OpenJDK
- Apache NetBeans IDE

35

Robodata & RobodataCUI Code Repositories

- Robodata including PCAPNG transformation: https://gitlab.nps.edu/Savage/robodata
- RobodataCUI including KLV transformation: https://gitlab.nps.edu/SavageDefense/robodatadefense

36