

TINE CORE MEETING

2.8.2018

4.6.3 Fixes and Features...

- In NotifyClient() (client.c) :
 - Unit stress test failing with too many link timeouts (TCP/STREAM mode)

```
client.c Revision 4079
4720 ... c->blcounter = 0;
4721 ..}
4722 .. bmode = BASEMODE(c->mode); /* base link mode */
4723 .. fireCallback = (cbfcn == NULL) ? c->cancelPending : c->suppressCallback;
4724 .. if (isGrouped && lgroupCanNotify(c,&cc)) resetPending = fireCallback = F;
4725 .. LastCompletionDataType = c->formatOut;
4726 .. LastCompletionDataSize = c->lastDataSize;
4727 .. LastIncomingLinkId = i;
4728 .. gDataTimeStamp = PutDataTimeStampU(0,c->dtimestamp,c->dtimestampUSEC);
4729 /* call the primary callback */
4730 if (fireCallback)
4731 {
4732 ... if (isTrace)
4733 ... {
4734 ... char txt[256];
4735 ... sprintf(txt,"needs to notify with status %d",cc);
4736 ... if (conTbl[i]->mcaIndex > 0) strcat(txt," (via MCA parent)");
4737 ... trceLinkOutput(i,"fireCallback",txt);
4738 ... }
4739 ... if (tineDebug > 2) dbglog("%.192s : fire callback",c->key);
4740 ... c->lastStatusReported = cc;
4741 ... if (cc == 0 && c->dataStatus == out_of_tolerance) cc = CE_SENDDATA + o
4742 ... TCBCALL(c,cc);
4743 ... if (c->bufCmp != NULL) memcpy(c->bufCmp,c->dataOut,c->sizeBytesOut);
4744 ..}
4745 if (bmode <= CH_SINGLE) c->mode = CH_CANCEL; /* can't do this often enough
4746 .. if (c->queueLen > 0) c->queueLen--; /* reduce the queue length -> usually
4747 .. /* continue checking for dependencies and queues ... */
4748 # if defined(SMALL_TIME_LIB)
4749 .. if (lc->hasNotifiedOnce && gAutoLinkWatchdogs) attachLwdLink(i);
4750 # endif
4751 .. hasData = HAS_DATA(cc);
4752 .. if (hasData) c->hasNotifiedOnce = TRUE;
4753 .. if (isBlackListed) c->linkStatus = link_blacklisted;
4754 # ifndef SMALL_TIME_LIB
...
LastCompletionDataType = c->formatOut;
LastCompletionDataType = c->formatOut;

client.c : Working Copy
4720 ... c->blcounter = 0;
4721 ..}
4722 .. bmode = BASEMODE(c->mode); /* base link mode */
4723 .. fireCallback = (cbfcn == NULL) ? c->cancelPending : c->suppressCallback;
4724 .. if (isGrouped && lgroupCanNotify(c,&cc)) resetPending = fireCallback = F;
4725 .. LastCompletionDataType = c->formatOut;
4726 .. LastCompletionDataSize = c->lastDataSize;
4727 .. LastIncomingLinkId = i;
4728 .. gDataTimeStamp = PutDataTimeStampU(0,c->dtimestamp,c->dtimestampUSEC);
4729 /* call the primary callback */
4730 if (fireCallback)
4731 {
4732 ... if (isTrace)
4733 ... {
4734 ... char txt[256];
4735 ... sprintf(txt,"needs to notify with status %d",cc);
4736 ... if (conTbl[i]->mcaIndex > 0) strcat(txt," (via MCA parent)");
4737 ... trceLinkOutput(i,"fireCallback",txt);
4738 ... }
4739 ... if (tineDebug > 2) dbglog("%.192s : fire callback",c->key);
4740 ... c->lastStatusReported = cc;
4741 ... if (cc == 0 && c->dataStatus == out_of_tolerance) cc = CE_SENDDATA + o
4742 ... TCBCALL(c,cc);
4743 ... if (c->bufCmp != NULL) memcpy(c->bufCmp,c->dataOut,c->sizeBytesOut);
4744 ..}
4745 if (bmode <= CH_SINGLE) c->mode = CH_CANCEL; /* can't do this often enough
4746 .. if (c->queueLen > 0) c->queueLen--; /* reduce the queue length -> usually
4747 .. /* continue checking for dependencies and queues ... */
4748 # if defined(SMALL_TIME_LIB)
4749 .. if (lc->hasNotifiedOnce && gAutoLinkWatchdogs) attachLwdLink(i);
4750 # endif
4751 .. hasData = HAS_DATA(cc);
4752 .. if (hasData) c->hasNotifiedOnce = TRUE;
4753 .. if (isBlackListed) c->linkStatus = link_blacklisted;
4754 # ifndef SMALL_TIME_LIB
...
LastCompletionDataType = c->formatOut;
LastCompletionDataType = c->formatOut;
```

4.6.3 Java

- Local History relative tolerance.

```
6361 class hstTolHndlr implements csvHandler
6362 {
6363     private hstRowHndlr rHndlr;
6364     hstTolHndlr(hstRowHndlr rowHndlr) { rHndlr = rowHndlr; }
6365     public int process(String strValue,int index)
6366     {
6367         int i=0;
6368         try
6369         {
6370             if ((i=strValue.indexOf('%')) == -1)
6371             { // absolute tolerance !
6372                 rHndlr.aTolerance = Float.parseFloat(strValue);
6373                 rHndlr.pTolerance = 0;
6374             }
6375             else
6376             {
6377                 rHndlr.aTolerance = 0;
6378                 rHndlr.pTolerance = Float.parseFloat(strValue.substring(0,i));
6379                 if (rHndlr.pTolerance >= 1) rHndlr.pTolerance /= 100.0;
6380             }
6381         }
6382         catch (NumberFormatException e)
6383         {
6384             rHndlr.pTolerance = 0;
6385             rHndlr.aTolerance = 0;
6386         }
6387         return 0;
6388     }
6389 }
```



4.6.3 Java

- TLinkFactory.userType (String)
 - In advance of Release 5.0 :

```
public boolean isDoocsClient()
{
    if (userType == null) return false;
    if (userType.compareToIgnoreCase("JDDD") == 0) return true;
    if (userType.contains("DOOCS")) return true;
    return false;
}
```

- => jddd Workarounds only apply to actual jddd clients !

Odds and Ends

- Tutorials
 - Interest in tutorials (Python, Acop.NET, ?)
- Acop.NET News
 - AcopForm, 3-D charts, etc.
 - PCaPAC presentation as early alpha @ Wednesday meeting in a few weeks; as beta @ TINE Users Meeting in September ...
- Central Archive News:
 - Master/Slave role reversals ...
 - PETRA, XFEL
 - Coming soon: anybody else?

Release 5.0

- Release 5.0 ...
 - Preliminary versions (C and Java) in SVN.
 - C Unit Client and Servers tested (with even more tests!)
 - R5 server – R5 client
 - R5 server – R4 client
 - R4 server – R5 client
 - Java server and clients tested
 - No 'java unit server'
 - Numerous unit tests by hand ...

Release 5.0 Transport headers

```
]typedef struct /* struct is passed over the net (legacy) */
{
    UINT16 totalSize; /**< total packet size in bytes (must match the bytes read) */
    char userName[USERNAME_SIZE]; /**< caller name */
    short tineProtocol; /**< tine protocol level (modern: 6 for contracts, 4 for globals) */
    UINT16 number; /**< tine version number (contracts) or number of incoming keywords (globals) */
} PktHdr;
#define PKTHDR_SIZE 22

]**
* \internal
* \brief The top level header which defines the incoming request (or globals) data packet.
*
*/
]typedef struct PayloadHeader/* struct is passed over the net (modern, little endian) */
{
    UINT32 totalSizeInBytes; /**< total packet size in bytes (must match the bytes read) */
    UINT32 headerSizeInBytes; /**< size of this header in bytes (allows for dynamic extension ?) */
    UINT32 pid; /**< process ID of caller */
    SINT32 tineVersion; /**< encoded specific version information */
    SINT16 numberSegments; /**< number of following segments (contract requests, globals keywords) */
    SINT16 tineProtocol; /**< tine protocol level (modern: 7) :: still at position 18 */
    UINT16 endianness; /**< endianness of caller : 0 or 1 (little or big) -> all subsequent header
    UINT16 encoding; /**< character encoding: ascii, utf-8, uni-code ? -> requires byte swapping
    char userName[USERNAME_SIZE]; /**< caller name */
    char userType[USERTYPE_SIZE]; /**< application type tag : 8 characters */
} ReqHdr, GlbHdr;
```

Release 5.0 Transport headers

- Response (Server -> Client)
 - Legacy: just single unsigned short with sizeInBytes ...
 - Modern:

```
/**
 * \internal
 * \brief The top level header which defines the incoming response data packet.
 *
 */
typedef struct
{
    SINT32 tineProtocol; /**< incoming tine protocol as integer -> distinct from legacy response of total
    UINT32 totalSizeInBytes; /**< total packet size in bytes (must match the bytes read) */
    UINT32 headerSizeInBytes; /**< size of this header in bytes (allows for dynamic extension ?) */
    SINT16 numberSegments; /**< number of following segments (contract responses) */
    UINT8 endianness; /**< endianness of FEC : 0 or 1 (little or big) -> all subsequent headers and data
    UINT8 encoding; /**< character encoding: ascii, utf-8, uni-code ? -> requires byte swapping */
    char fecName[USERNAME_SIZE]; /**< FEC name */
} RspHdr;
#define RSPHDR_SIZE 32
```


Release 5.0 Transport headers

- Handling legacy responses :

```
int prepIncomingResponseHdr(RspHdr *hdr, RspSpecs *spc, BYTE *stream, UINT32 totalSizeInBytes)
{
    UINT32 p = PeekDWord(*(UINT32 *)stream);
    if (p > DEFAULT_TRANSPORT_PROTOCOL_LEVEL)
    { /* indicative of legacy system (pre Release 5); 1st 2 bytes contain total size */
        p = PeekWord(*(UINT16 *)&stream[PHDR_PROT_OFFSET+2]) == 6 ? 6 : 5;
    }
    if (p < MINIMUM_REQHDR_PROTOCOL)
    { /* map to modern request header */
        ConTblEntry *c = NULL;
        int id = -1;
        UINT16 hsiz = 2;
        hdr->tineProtocol = p;

        if (totalSizeInBytes == 0)
        { /* normal (reassembly) case */
            totalSizeInBytes = (UINT32)PeekWord(*(UINT16 *)stream);
        }
        else
        { /* stream case (it's all in) */
            UINT32 msgsiz = PeekDWord(*(UINT32 *)stream);
            if (msgsiz > totalSizeInBytes) totalSizeInBytes = msgsiz;
            hsiz = 2; /* this is artificially still 2 ! */
        }
        hdr->totalSizeInBytes = totalSizeInBytes;
        hdr->endianness = BO_LITTLE_ENDIAN;
    }
}
```

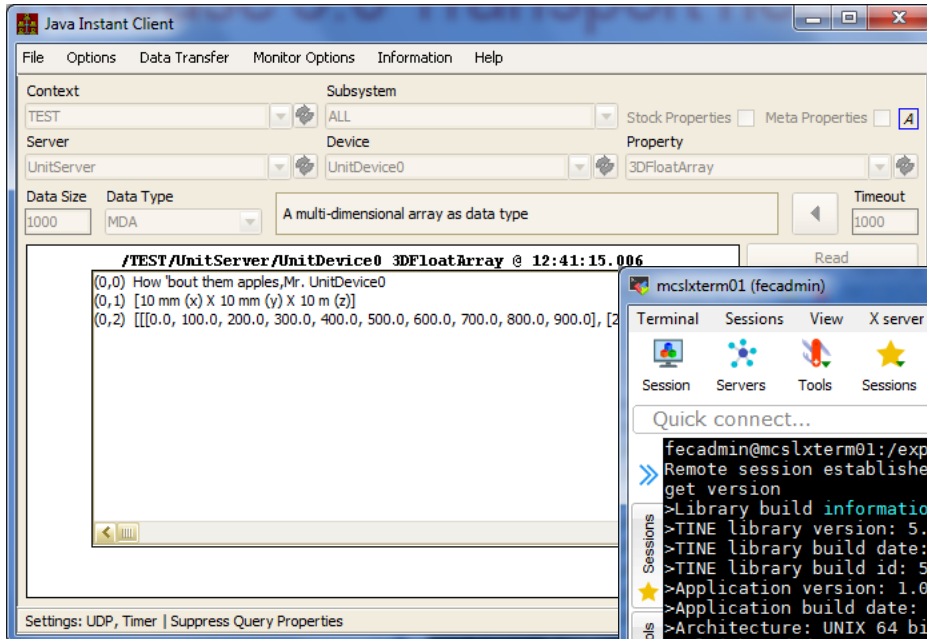
Release 5.0 Transport headers

- Subscription header(s) follow the Response header:

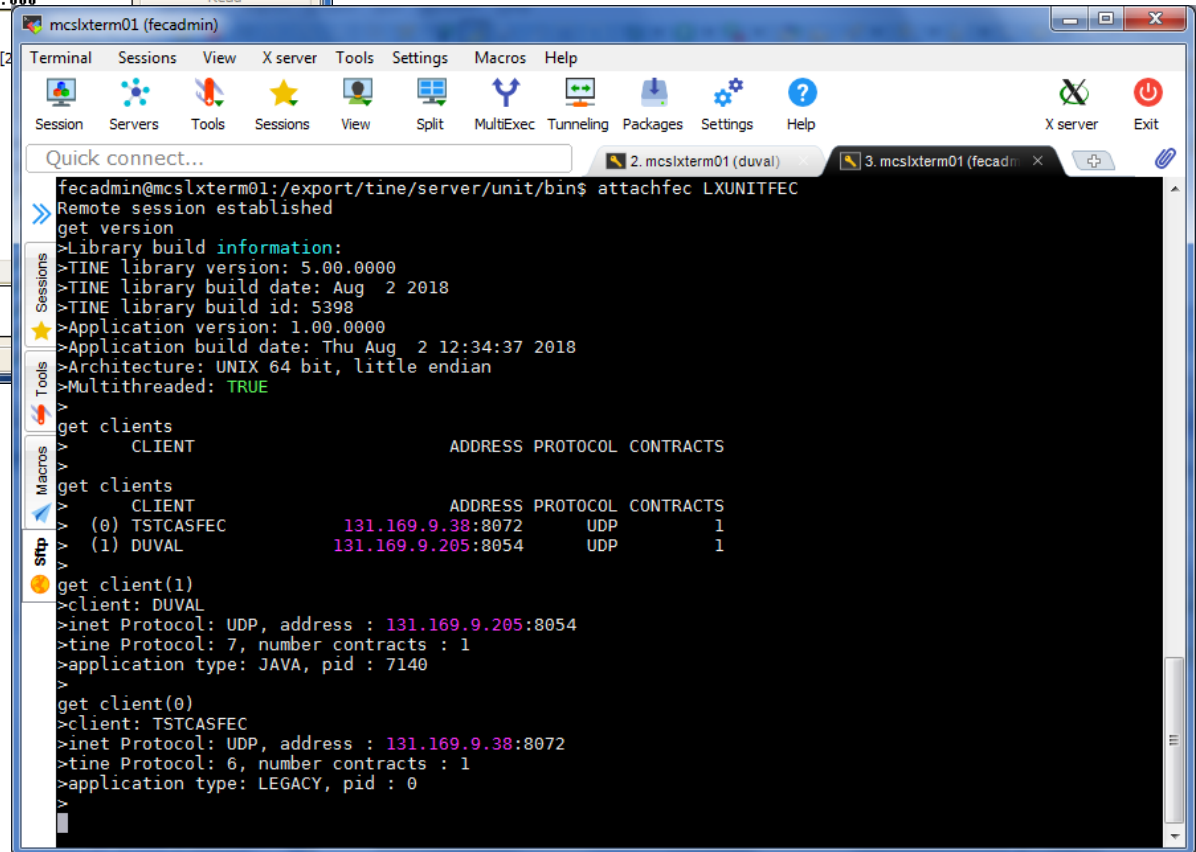
```
/**
 * \internal
 * \brief subscription response header struct (reply to client)
 *
 * modern : release 5.xx era
 * msgsize and mtu are now UINT32
 * separate fields for returned data size in bytes and size in elements
 * struct is passed over the net : tineProtocol 7 (out)
 */
typedef struct
{
    UINT32 msgSize;           /**< this message size in bytes */
    UINT16 subId;            /**< supplied by the consumer upon registration (con tbl id) */
    UINT16 retCode;          /**< return completion code from the eqm */
    UINT16 numBlks;          /**< total number of message blocks */
    UINT16 blkNum;           /**< block number of this message */
    UINT32 mtu;              /**< maximum transport unit */
    UINT16 dataFormat;       /**< format of contract data */
    UINT16 counter;          /**< subscription counter (how much is left) */
    UINT16 tineProtocol;     /**< tine protocol level */
    UINT16 xferReason;       /**< transfer reason flag (one of the CX_ codes above) */
    UINT32 starttime;        /**< supplied by the consumer upon registration (@ byte 20)*/
    UINT16 stssize;          /**< size of error/status string space */
    UINT16 stscode;          /**< access status code */
    UINT32 timestamp;        /**< data timestamp (UTC wraps in 2038) */
    UINT32 timestampUSec;    /**< usec fraction of data timestamp */
    UINT32 userstamp;        /**< application setable data stamp */
    UINT32 sysstamp;         /**< systematic data stamp (wraps every 13 years @ 10 Hz continuous */
    UINT32 dataSize;         /**< size of contract data */
    UINT32 msgSizeInElements; /**< number of elements (of type format) returned in this message */
} SubRspHdr;
#define SUBRSPHDR_SIZE (9*4 + 10*2) /* 56 bytes */
```

sysstamp and userstamp still 32-bit integer and NOT 64-bit integer.

Release 5.0 Transport headers

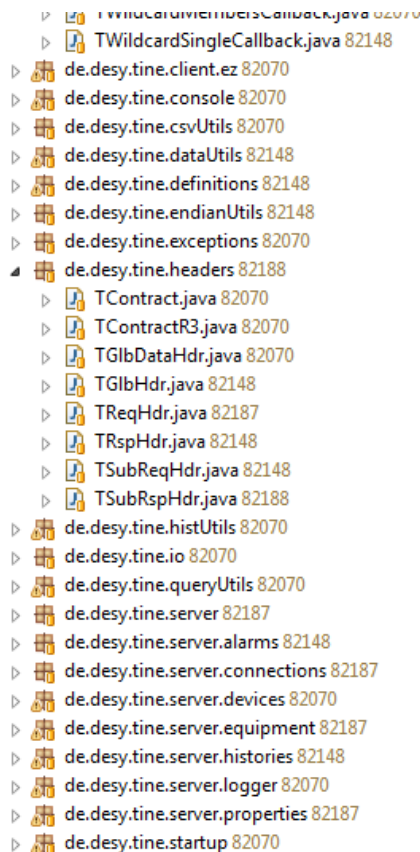


Release 5 java client
Release 5 C server ...



Release 5.0 Transport headers

- Java: same transport headers and compatibility must be present!



```
105 public byte[] toByteArray()
106 { // prepare outgoing bytestream (e.g. client sending request to server)
107     totalSizeInBytes = (int)(msgSizeInBytes + getHeaderSizeInBytes());
108     try
109     {
110         dBuffer.reset();
111         DataOutputStream ds = new DataOutputStream(dBuffer);
112         if (tineProtocol > TTransport.LegacyProtocolLevel)
113         {
114             ds.writeInt(Swap.Int(totalSizeInBytes));
115             ds.writeInt(Swap.Int(headerSizeInBytes));
116             ds.writeInt(Swap.Int(pid));
117             ds.writeInt(Swap.Int(tineVersion));
118             ds.writeShort(Swap.Short(numberSegments));
119             ds.writeShort(Swap.Short(tineProtocol));
120             ds.writeShort(Swap.Short(endianness));
121             ds.writeShort(Swap.Short(encoding));
122             Arrays.fill(bstr, (byte)0);
123             byte[] tmp = username.getBytes();
124             for (int i=0; i<16 && i<tmp.length; i++) bstr[i] = tmp[i];
125             ds.write(bstr,0,16);
126             Arrays.fill(bstr, (byte)0);
127             tmp = usertype.getBytes();
128             for (int i=0; i<8 && i<tmp.length; i++) bstr[i] = tmp[i];
129             ds.write(bstr,0,8);
130         }
131     }
132     else
133     { // legacy server
134         ds.writeShort(Swap.Short(totalSizeInBytes));
135         Arrays.fill(bstr, (byte)0);
136         byte[] tmp = username.getBytes();
137         for (int i=0; i<16 && i<tmp.length; i++) bstr[i] = tmp[i];
138         ds.write(bstr,0,16);
139         ds.writeShort(Swap.Short(tineProtocol));
140         ds.writeShort(Swap.Short(revisionId));
141     }
142 }
```

Release 5.0 Transport headers

- Other embellishments ...
- C side:
 - Many macro `#defines` (which have been around since ‘the beginning’) have moved to `enums`
 - Many macros -> functions
 - Forget about being nice to MS-DOS !

```
/** \def Error Number Definitions */
enum Errors
{
    operation_success           = 0,  /**< successful completion */
    illegal_line                = 1,  /**< illegal line number */
    illegal_format              = 2,  /**< illegal format or data type */
    illegal_arithmetic          = 3,  /**< illegal arithmetic expression */
    ambiguous                   = 4,  /**< ambiguous input information */
    illegal_delimiter           = 5,  /**< illegal delimiter in expression */
    zero_divide                 = 6,  /**< divide by zero */
    work_area_full              = 7,  /**< working area buffer is full */
    non_existent                = 8,  /**< requested target does not exist */
    invalid_transport           = 9,  /**< transport medium is not valid */
    data_size_mismatch          = 10, /**< incoming data payload larger than requested ? */

    no_data_in_range            = 11, /**< no data found in input range */
    un_allocated                 = 12, /**< requested target is not allocated in the given cor
    no_such_line                 = 13, /**< expected line not found in database */
    illegal_data_size           = 14, /**< give data size is illegal or not valid */
    io_error                     = 15, /**< bus or network io error */
    illegal_context              = 16, /**< given context is illegal */
    runtime_error                = 17, /**< runtime error */
    system_error                 = 18, /**< system internal error */
    hardware_busy                = 19, /**< hardware is busy */
    argument_list_error          = 20, /**< input argument or parameter is invalid (alias for
    invalid_parameter            = 20, /**< input argument or parameter is invalid */

    file_error                   = 21, /**< file access error */
    use_stream_transport         = 22, /**< datagram message size exceeds the allowable mtu (r
```

Release 5.0 Transport headers

- C Side:
 - No more clashes with STL or MFC headers !
 - Works with or without namespace wrapper

```
namespace tne  
{  
    #include "tne.h"  
}
```

Release 5.0 Transport headers

- C and Java:
 - New format types for strong typing:
 - CF_UINT16
 - CF_UINT32
 - CF_UINT64
 - Auto-grow the connection table by default.
 - Not yet making use of the endianness or encoding information.