TINE CORE MEETING

17.4.2018

4.6.3

- Ipv6 residue ...

Modern Windows, Linux, MACOS know about option IPV6_V6ONLY

```
# if defined(FIX_IPV6_V6ONLY)
  if (family == AF_INET6)
  {
    optval = 0; optlen = sizeof(optval);
    if (setsockopt(*s,IPPROTO_IPV6,IPV6_V6ONLY,(char*)&optval,optlen)) soerr("IPV6_V6ONLY");
  }
# endif
  if (isMCastPort(port)) reuseFlg = TRUE;
  if (reuseFlg)
  {
    optval = 1; optlen = sizeof(optval);
    if (setsockopt(*s,SOL_SOCKET,SO_REUSEADDR,(char *)&optval,optlen)) soerr("REUSEADDR");
#   ifdef MACOS
    optval = 1; optlen = sizeof(optval);
    if (setsockopt(*s,SOL_SOCKET,SO_REUSEPORT,(char *)&optval,optlen)) soerr("REUSEPORT");
#   endif
  }
# if defined(UNIX) || defined(WIN32)
  if (rcvbuf > 0)
```

```
#if defined(USE_IPV6) && !defined(NO_IPV6_V6ONLY)
# define FIX_IPV6_V6ONLY
#endif
```

In prolog.h

legacy Windows, Solaris, VxWorks …

4.6.3 Fixes and Features…

- Some memory leaks …

```
void clearDependentLinks(int linkId)
{
  ConTblEntry *c;
  DpdTblEntry *lnk, *nlnk=NULL;
  if (linkId < 0 || linkId > nConnectionTableEntries) return; /* not a valid link table entry ! */
  c = conTbl[linkId];
  for (lnk=c->dpdLink; lnk != NULL; lnk=nlnk)
  {
    nlnk = lnk->nxt;
    free(lnk);
  }
  c->dpdLink = NULL;
}
void removeDependentLink(int linkId)
{
  ConTblEntry *c;
  DpdTblEntry *lnk;
  int parentId, found=FALSE;
  char key[256];
  if (linkId < 0 || linkId > nConnectionTableEntries) return; /* not a valid link table entry ! */
  parentId = conTbl[linkId]->boundToId;
```

Certain scenarios involving dependent links …
(multiple calls to the ‘same thing’)

4.6.3 Fixes and Features…

- In int getFormattedHistoryDataPoint(DTYPE *dst,DTYPE *src)
  - -> missing data type CF_INT64 (Arthur):

```
switch (fmt)
{ /* requested data format */
  case CF_FLOAT:
    return GetValuesAsFloat(src,(float *)data,1);
  case CF_INT32:
    return GetValuesAsLong(src,(SINT32 *)data,1);
#   ifdef SINT64
  case CF_INT64:
    return GetValuesAsDLong(src,(SINT64 *)data,1);
#   endif
  case CF_DOUBLE:
    return GetValuesAsDouble(src,(double *)data,1);
#   ifndef SMALL_TINE_LIB
  case CF_NAME16:
  case CF_NAME32:
  case CF_NAME48:
  case CF_NAME64:
```

4.6.3 Fixes and Features…

- CF_INT16 etc. vs. CF_UINT16 etc. ?
  - As yet: TINE like Java -> no explicit ‘unsigned’
    - Interpret as you like …
  - Utilities from toolkit.c :

```
TINE_EXPORT int GetValuesAsByte(DTYPE *d,BYTE *bval,int num) { ... }
TINE_EXPORT int GetValuesAsShort(DTYPE *d,short *sval,int num) { ... }
TINE_EXPORT int GetValuesAsFloat(DTYPE *d,float *fval,int num) { ... }
TINE_EXPORT int GetValuesAsLong(DTYPE *d,SINT32 *lval,int num) { ... }
TINE_EXPORT int GetValuesAsUnsignedLong(DTYPE *d,UINT32 *lval,int num) { ... }
#ifdef SINT64
TINE_EXPORT int GetValuesAsDLong(DTYPE *d,SINT64 *llval,int num) { ... }
TINE_EXPORT int GetValuesAsUnsignedDLong(DTYPE *d,UINT64 *llval,int num) { ... }
#endif /* SINT64 */
TINE_EXPORT int GetValuesAsDouble(DTYPE *d,double *dval,int num) { ... }
```

  - Should we introduce explicit ‘strong’ format identifiers ?
    - CF_UINT16, CF_UINT32, CF_UINT64
    - useful in property registration ?

Infinity …

- Central Archive Server
  - Searching for a value …

```
if (isinf(dt0) || isnan(dt0)) ccerr(file_error);
if (isinf(dt2) || isnan(dt2)) ccerr(file_error);
while (p2 > p0 + recsize)
{
  if (ts > (dt1=dt2)) { p1 = p2; break; }
  while (dt1 != dt0)
  { /* seek <-- */
    f = ((ts-dt0)/(dt1-dt0))*((double)(p2-p0)/recsize);
    if ((p1=p0+recsize*(long)f) >= p2) break;
    lseek(hFile,p1,SEEK_SET);
    read(hFile,hdr,hdrsiz);
    if (!useMinimalStorage)
    {
      dt1 = *(double *)hdr;
    }
    else
    {
      t1 = *(SINT32 *)hdr; dt1 = t1;
    }
    if (isinf(dt1) || isnan(dt1)) ccerr(file_error);
    if (dt1 < ts) break;
    p2 = p1;
    dt2 = dt1;
    it++;
  }
  dt0 = dt1; p0 = p1;
  while (dt2 != dt1)
  { /* seek --> */
    f = ((dt2-ts)/(dt2-dt1))*((double)(p2-p0)/recsize);
    if ((p1=p2-recsize*(long)f) <= p0) break;
    lseek(hFile,p1,SEEK_SET);
    read(hFile,hdr,hdrsiz);
    if (!useMinimalStorage)
    {
      dt1 = *(double *)hdr;
    }
    else
    {
```

read timestamp *dt2* as ‘inf’ (i.e. 0x7ff000000)

gets lost in lseek() !

4.6.3 Fixes and Features…

- Include same check in local history routines …
  - Modern OSes know about ‘isinf(v)’ and ‘isnan(v)’ but of course some don’t …
    - Older Windows (use ‘_isnan(v)’ and ‘!isfinite(v)’)
    - Solaris (use ‘!isfinite(v)’)
    - VxWorks doesn’t need it ! (no file system)

Bug-fix of the month …

```
TINE_EXPORT int SetConnectionTableCapacity(int value)
{ /* ConTblCapacity established following allocation */
  int cc = 0;
  if (value < 10) return invalid_data;
  if (nConnectionTableEntries > 0)
  {
    feclogEx(TINE_LOGLEVEL_ERROR,"Client Connection Table already in use : reallocation not allowed");
    return already_assigned;
  }
  if (WaitForSystemMutex(hLinkTblMutex,-1) != 0) return mutex_error;
  if (value < gConTblFloor) value = gConTblFloor;
  if (value > MaxNumConnections)
  {
    if (conTbl != NULL)
    {
      ConTblEntry **newTbl = (ConTblEntry **)realloc(conTbl,value*sizeof(ConTblEntry *));
      if (newTbl != NULL)
      {
        memset(&newTbl[MaxNumConnections],0,(value-MaxNumConnections)*sizeof(ConTblEntry *));
        conTbl = newTbl;
        feclog("Client Connection Table has be reallocated for %d entries",value);
      }
      else
      {
        feclogEx(TINE_LOGLEVEL_ERROR,"Client Connection Table reallocation for %d entries failed",value);
        value = MaxNumConnections;
        cc = out_of_client_memory;
      }
    }
  }
  ConTblCapacity = MaxNumConnections = value;
  ReleaseSystemMutex(hLinkTblMutex);
  return cc;
}
```

new!

can be problematic !

Bug-fix of the month …

Work-around solution for Lars :

```
TINE_EXPORT void SetConnectionTableMinimumCapacity(int value)
{
  gConTblFloor = value;
}
```

or :

```
if ((ptr=getenv("TINE_CONNECTION_TABLE_MIN_CAPACITY")) != NULL)
{ /* only make use of the env if not set via API */
  if (gConTblFloor == 0) gConTblFloor = atoi(ptr);
}
```

Table capacity no allowed to be less than gConTblFloor !

Bug-fix of the month before …

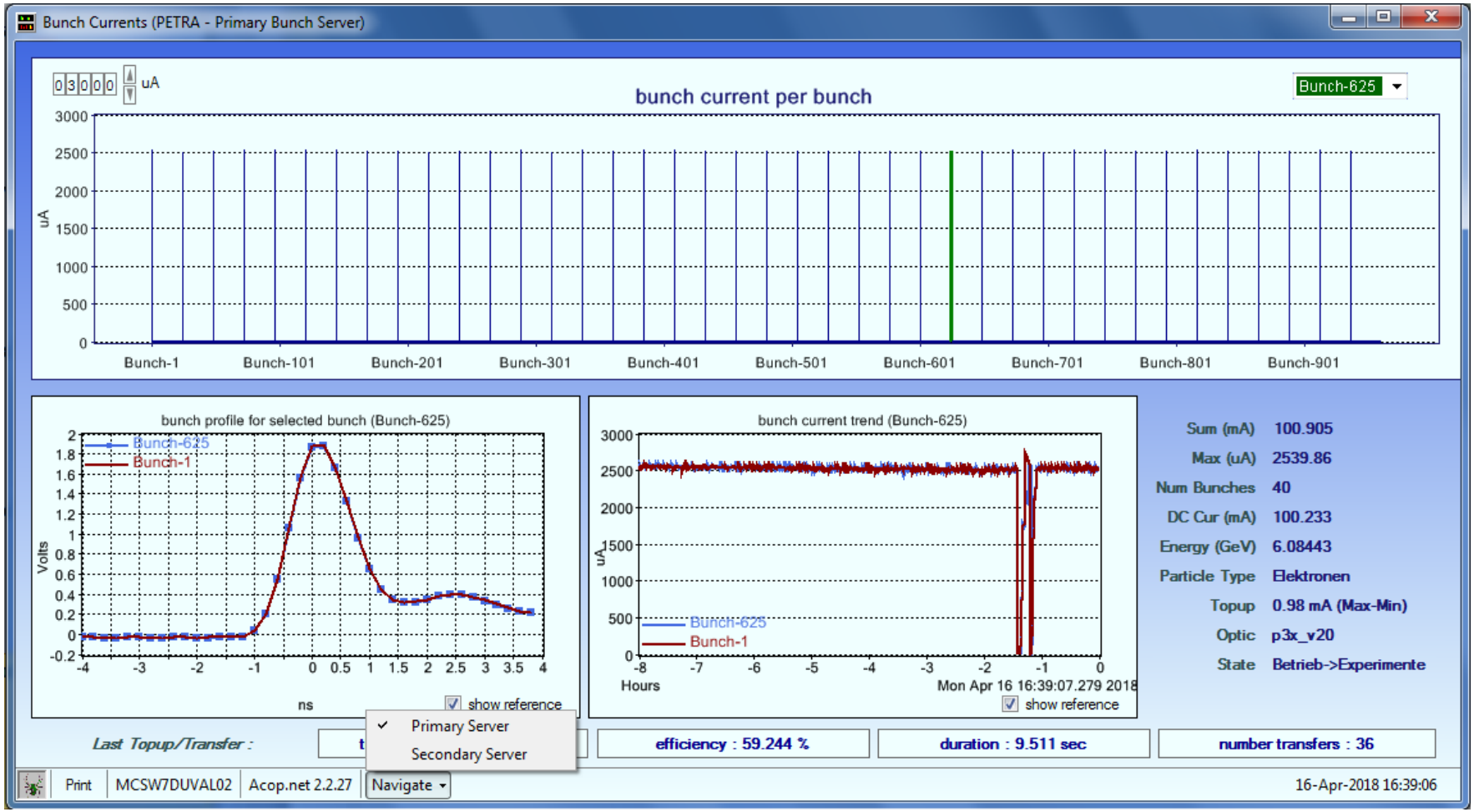
- Thanks to an Acop.NET application …
  - CloseLink(int linkId)
  - Scenario:

- Component **A** has a real MCA link over all devices (e.g. bunch currents) and uses a *bound data reference.*
- Component **B** has a single-element of same MCA link (e.g. a trend of one of the bunches)

- Either **B** owns the MCA link and **A** attaches to **B** or **A** owns the link and **B** attaches to **A**

- **A** Detaches its link and frees its data.
- **B** crashes and burns … (if **A** owns the link!)

Bug-fix of the month before …



Bug-fix of the month before …

- If the ‘owner’ closes and is using its own memory then allocate a parallel data set !
  - inside _closeLink() :

```
if (c->dpdLink != NULL)
{ /* parent link is canceled with dependencies ! */
  c->cancelPending = TRUE;
  c->needsNotification = FALSE;
  if (c->allocatedBytes == 0)
  { /* must re-shuffle the data storage as the caller might free his memory! */
    if ((c->dataOut=(char *)SystemCalloc(1,(size_t)c->sizeBytesOut)) != NULL)
    {
      c->allocatedBytes = c->sizeBytesOut;
    }
  }
  goto out;
}
```

4.6.3 and IPv6

- Both C and Java code are IPv6 ready!
- Acop.NET news ?
  - New bells and whistles …
  - 3-D plots in AcopChart?

Release 5.0

- Release 5.0 …
  - http://adweb.desy.de/mcs/TINE_Presentations/CollaborationMeeting2012/CollaborationMeeting2012.pdf
  - Begin with release 5 headers …
    - extended with more information
  - String sets: utf8
    - applies to ‘names, strings that are seen’.
  - Full compatibility with release 4
    - worry about release 3?
  - Queried and registered information …
    - need extensions?
    - e.g. a lot of (rather fuzzy) discussions about data mining etc. => property registration to contain more attributes?
      - e.g. ‘IsDataMineable’

Release 5.0

- General To-Do items ?
  - Automatic overloading of some data types?
  - e.g. (one of Marcus' 1st servers …)

phaseSOLL is only registered to deliver data type **FLTINT** (a data – status doublet)

If the server does not itself overload the property to handle **FLOAT** we could automatically allow such a call …