TINE CORE MEETING

25.5.2016

Noteworthy C-Lib Changes

- Refactoring (thanks to TANGO)
  - Use gIsCyclePollingMode instead of ‘POLLING’, ‘SELECTING’. e.g.

```
if (ServerCycleMode == POLLING && soioctl(sck,FIONBIO,(char *)&mode))
  soerr("NONBLOCKING");
```

  - becomes

```
if (gIsCyclePollingMode && soioctl(sck,FIONBIO,(char *)&mode)) soerr("NONBLOCKING");
```

- linkTbl mutex protection inside checkConnections().

```
c->xferReason = CX_NULL;
c->suppressCallback = FALSE;
mode = c->mode; /* note existing transport mode */
if (!strcmp(FecTbl[c->fecIdx].fecName,"FECSINK")) cc = address_unknown;
ReleaseSystemMutex(hLinkTblMutex); /* never know what someone might do inside a callback ... */
NotifyClient(i,cc);
if (WaitForSystemMutex(hLinkTblMutex,gSystemTick) != 0) return;
bmode = BASEMODE(c->mode); /* check again in case it was canceled inside the callback ... */
if (c->bytesin > 0) nPartialTransfers++;
```

Noteworthy C-Lib Changes

- linkTbl mutex protection …

```
Thread 2:: RcvGlb
0  libsystem_kernel.dylib      0x00007fff9862f2a2 poll + 10
1  libDOOCSapi.18.10.5.dylib   0x00000001060cfe75 glbRecvThreadTask + 197
2  libsystem_pthread.dylib     0x00007fff9abfd99d _pthread_body + 131
3  libsystem_pthread.dylib     0x00007fff9abfd91a _pthread_start + 168
4  libsystem_pthread.dylib     0x00007fff9abfb351 thread_start + 13

Thread 3 Crashed:: tcycler
0  libDOOCSapi.18.10.5.dylib   0x000000010604f2ee checkConnections + 1070
1  libDOOCSapi.18.10.5.dylib   0x000000010604a743 clientCycle + 339
2  libDOOCSapi.18.10.5.dylib   0x00000001060d35ee _SystemCycle + 334
3  libtinemt.dylib             0x00000001063a45e6 cycleTmrTask + 102
4  libtinemt.dylib             0x00000001063a46e8 cycleTmrThread + 136
5  libsystem_pthread.dylib     0x00007fff9abfd99d _pthread_body + 131
6  libsystem_pthread.dylib     0x00007fff9abfd91a _pthread_start + 168
7  libsystem_pthread.dylib     0x00007fff9abfb351 thread_start + 13

Thread 4:: clnt_update
0  libsystem_kernel.dylib      0x00007fff9862e10a __semwait_signal + 10
1  libsystem_c.dylib           0x00007fff892f5d17 nanosleep + 199
2  libDOOCSapi.18.10.5.dylib   0x0000000106021bc3 update_thread + 115
3  libsystem_pthread.dylib     0x00007fff9abfd99d _pthread_body + 131
4  libsystem_pthread.dylib     0x00007fff9abfd91a _pthread_start + 168
5  libsystem_pthread.dylib     0x00007fff9abfb351 thread_start + 13
```

```
Thread 5:: clnt_sl_update
0  libsystem_kernel.dylib      0x00007fff9862ddb6 __psynch_cvwait + 10
1  libsystem_pthread.dylib     0x00007fff9abfe728 _pthread_cond_wait + 767
2  libDOOCSapi.18.10.5.dylib   0x00000001060224b4 eq_res::wait_slow(int*) + 84
3  libDOOCSapi.18.10.5.dylib   0x000000010602243b update_slow + 43
4  libsystem_pthread.dylib     0x00007fff9abfd99d _pthread_body + 131
5  libsystem_pthread.dylib     0x00007fff9abfd91a _pthread_start + 168
6  libsystem_pthread.dylib     0x00007fff9abfb351 thread_start + 13

Thread 6:: clnt_cycler
0  libsystem_kernel.dylib      0x00007fff9862e10a __semwait_signal + 10
1  libsystem_c.dylib           0x00007fff892f5d17 nanosleep + 199
2  libDOOCSapi.18.10.5.dylib   0x00000001060d359d _SystemCycle + 253
3  libDOOCSapi.18.10.5.dylib   0x000000010602d407 tine_cycle(int) + 23
4  libDOOCSapi.18.10.5.dylib   0x0000000106022552 cycle_thread + 66
5  libsystem_pthread.dylib     0x00007fff9abfd99d _pthread_body + 131
6  libsystem_pthread.dylib     0x00007fff9abfd91a _pthread_start + 168
7  libsystem_pthread.dylib     0x00007fff9abfb351 thread_start + 13
```

Notice anything ‘suspicious ?

Noteworthy C-Lib Changes

- avoid a potential deadlock when debugging is ON and there are send socket errors
  - also in this regard: treat 'printf' as a 'send' (as it is on VxWorks).

inside dbglog() :

```
    }
# elif defined(UNIX)
  for (i=0; i<ipcClnLstSize; i++) if (write(IPCfds[i],str,strlen(str)) < 0) perror("write");
# endif
# endif
  dbgprintf(str,use_printf);
err:
  ReleaseSystemMutex(hLogMutex);
  return cc;
```

dbgprintf() :

```
int dbgprintf(char *str,int pstdout)
{
  int i;
  if (WaitForSystemMutex(hSndMutex,gSystemTick) != 0) return semaphore_busy;
  if (pstdout) printf("%s",str);
  for (i=0; i<nDBGsck; i++) send(dbgSckTbl[i].sck,str,(int)strlen(str),0);
  ReleaseSystemMutex(hSndMutex);
  return 0;
}
```

Noteworthy C-Lib Changes

- VxWorks ‘task trace ….’

```
0x001f6858 select       +0x3e0: semTake ()
0x002848e0 semTake      +0xac : semBTake ()
value = 0 = 0x0
-> tt tSrvCycThr
0x0019f2d0 vxTaskEntry  +0x48 : srvThreadTask (0)
0x032e0534 srvThreadTask+0x68 : serverCycle ()
0x032779f8 serverCycle  +0x648: doScheduler ()
0x0327995c doScheduler  +0xaf0: doContract ()
0x032763a8 doContract   +0x1b74: DataDump (0x12e11118, 0x25ebe9c, 0xf)    <<<<<<<<<<<<<<< DEBUGGING ist ON !!!
0x032d78b4 DataDump     +0x78 : ttyoutput ()
0x033027fc ttyoutput    +0x94 : waitForMutex (0x30db140, 0xffffffff)      <<<<<<<<<<<<<<  aber er braucht den hLogMutex
0x032db344 waitForMutex +0xb4 : semTake ()
0x002848e0 semTake      +0xac : semBTake ()
value = 0 = 0x0
-> tt tSchdlrThr
0x0019f2d0 vxTaskEntry  +0x48 : schThreadTask (0)
0x032ddfd8 schThreadTask+0x3c : waitForSemaphore (0x12e1a5b0, 0xffffffff)
0x032db3ec waitForSemaphore+0x28 : waitForMutex (0x12e1a5b0, 0xffffffff)
0x032db344 waitForMutex +0xb4 : semTake ()
0x002848e0 semTake      +0xac : semBTake ()
value = 0 = 0x0
-> tt tSTKEQMthr
0x0019f2d0 vxTaskEntry  +0x48 : stkThreadTask ()
0x032df440 stkThreadTask+0x9c : waitForMutex (0x30e01f0, 0xffffffff)
0x032db344 waitForMutex +0xb4 : semTake ()
0x002848e0 semTake      +0xac : semBTake ()
value = 0 = 0x0
-> tt glbRecvThrea
C interp: unknown symbol name 'glbRecvThrea'.
-> tt tGlbRecv
0x0019f2d0 vxTaskEntry  +0x48 : glbRecvThreadTask ()
0x032df108 glbRecvThreadTask+0x170: getGlobalUpdate ()
0x03232a70 getGlobalUpdate+0xb00: dbglog ()                          <<<<<<<<<<< DEBUGGING ist ON!
0x033024c4 dbglog       +0xac : waitForMutex (0x30db140, 0xffffffff)    <<<<<<<<<<<<<<  aber er braucht den hLogMutex
0x032db344 waitForMutex +0xb4 : semTake ()
0x002848e0 semTake      +0xac : semBTake ()
value = 0 = 0x0
-> tt HISTORY
0x0019f2d0 vxTaskEntry  +0x48 : task_HISTORY ()
0x026fb030 task_HISTORY +0x150: _SystemScheduleProperty (0x2700178, 0x2700180)
0x032e4618 _SystemScheduleProperty+0x2c : _SystemSchedulePropertyEx ()
0x032e45a4 _SystemSchedulePropertyEx+0x69c: historyCycle ()
0x0325dbac historyCycle +0xf28: callEqm (0x3325824, 0x30d6fec, 0x33e06c0, 0x3325708, 0x3325738)
0x03272a74 callEqm      +0x8a0: dbglog ()                            <<<<<<<<<<<< DEBUGGING ist ON!
0x033026fc dbglog       +0x2e4: printf (0x3317aac, 0x30d6c90, 0x30d6caa, 0, 0, 0x2, 0x3310000, 0x30d6cbc)
                                                                     <<<<<<<<<<<< er hat den hLogMutex !!!!!!!!!!!!!!!
0x001cf088 printf       +0x88 : fioFormatV ()
0x001cef04 fioFormatV   +0x11a0: fioBufPrint ()
0x001cf64c fioBufPrint  +0x3c : 0x001f0264 ()
0x001f02b0 close        +0xc8 : 0x001f22bc ()
0x001f2354 ioswrite     +0x9c : 0x001f81ac ()
0x001f81dc tywrite      +0x34 : semTake ()
0x002848e0 semTake      +0xac : semBTake ()                          <<<<<<<<<<<< wir sind hier tief in VxWorks !!!
                                                                     <<<<<<<<<<<< Eventuell fehlt ein hSndMutex Schutz noch ?
```

Noteworthy C-Lib Changes

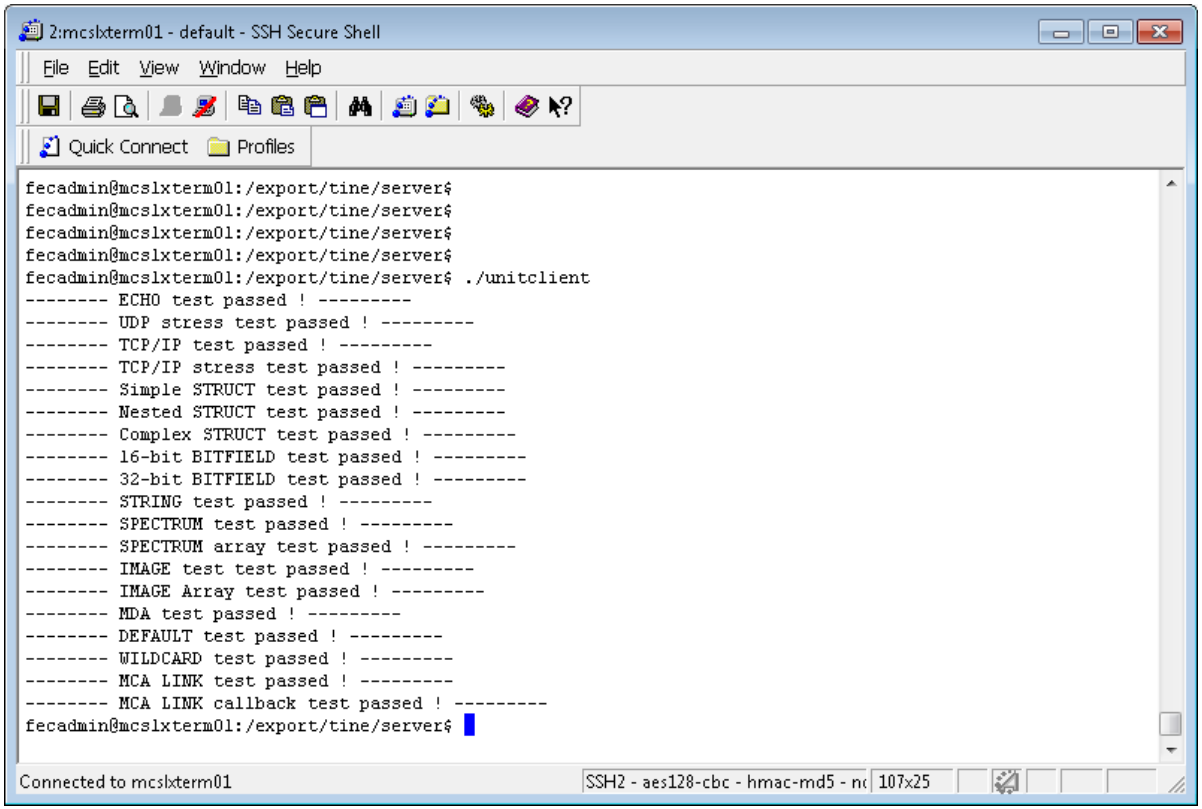
- introduce Set/GetSuspendCallbacks()
  - e.g. use around calls to ExecLink()
- added a 'flush address' and 'flush cache' to the console commands
  - To force server address reacquisition ‘now’ !
- fixed timereq = 0 in _attachLink so as to avoid long wait in establishing TCP connection

From prepSubRequest()

```
}
if (c->mode & CM_CONNECT && !fecIsLocal)
{
  if (inetProtocol == UDP || inetProtocol == IPX) continue;
  if (c->tcpSck == 0 && c->timereq == (UINT32)timestamp) continue; /* let a second go by between attempts */
  sck = getTcpConTblSocket(c,TCP);
  inetProtocol = TCP;
}
```

Noteworthy C-Lib Changes

- long wait in establishing TCP connection
  - Anywhere from 0 to 1000 msec !
  - Was not caught by ‘unit server/unit client’
  - Now it is!



Noteworthy C-Lib Changes

- respect the bound and mca links in getLinkIdFromCallbackId()

```
int getLinkIdFromCallbackId(int cbId)
{
  int i;
  for (i=0; i<nConnectionTableEntries; i++)
  {
    if (conTbl[i] == NULL) continue;
    if (!isActiveTransferMode(conTbl[i]->mode))
    { /* not an active link, but is it bound ? */
      if (conTbl[i]->boundToId <= 0 && conTbl[i]->mcaLink == NULL)
        continue;
    }
    if (conTbl[i]->wcLink != NULL &&
        conTbl[i]->wcLink->cbId == cbId)
    { /* a non-local wildcard link */
      return i;
    }
    else if (conTbl[i]->isWildcardLink)
    { /* otherwise a wildcard link */
      WcTblEntry *wc=conTbl[i]->wcLink;
      if (wc != NULL && wc->cbId == cbId) return wc->linkId;
    }
    else if (conTbl[i]->hasUserCallbackId &&
             conTbl[i]->cbId == (UINT32)cbId)
    { /* look for supplied callback id */
      return i;
    }
  }
  if (cbId >= 0 && cbId < nConnectionTableEntries && !conTbl[cbId]->hasUserCallbackId)
    return cbId;
  return -1;
}
```

Necessary addition !

Noteworthy C-Lib Changes

- Set/GetPutCommandsInCmdlog() to accompany the new ‘commands.log’
  - ‘commands.log’ in addition to ‘fec.log’
- add routine sckadrcmp() to replace a memcmp() on a comparison of SCKADR objects used in access lock comparison.

```
}
int isMemberLockSet(ExportListStruct *el,ClnHdr *cln)
{
  if (!el->accessLock.lockType) return TRUE; /* unlocked */
  if (strnicmp(el->accessLock.cln.usr,cln->usr,USERNAME_SIZE)) return FALSE;
  if (memcmp(&el->accessLock.cln.ipx,&cln->ipx,sizeof(IpxAdr)))
    return FALSE;
  if (sckadrcmp(&el->accessLock.cln.ip,&cln->ip)) return FALSE;
  return TRUE;
}
int isExclusiveProperty(ExportListStruct *el,char *prp)
```