TINE Release 4.0 News

(Oct 19, 2012: That was the month that was !)

“What a long, strange trip it’s been ….”

Release 4.3.2

- Embellishments and bug fixes (C Lib)
  - ExecLink("/LOCALHOST/eqm/…",…) now works !.
    - problem noticed by MSK !
    - a synchronous call 'in-process' (e.g. VxWorks
    - different from calling the eqm() routine explicitly!
      - checks credentials
      - can call meta properties, wildcards, etc.

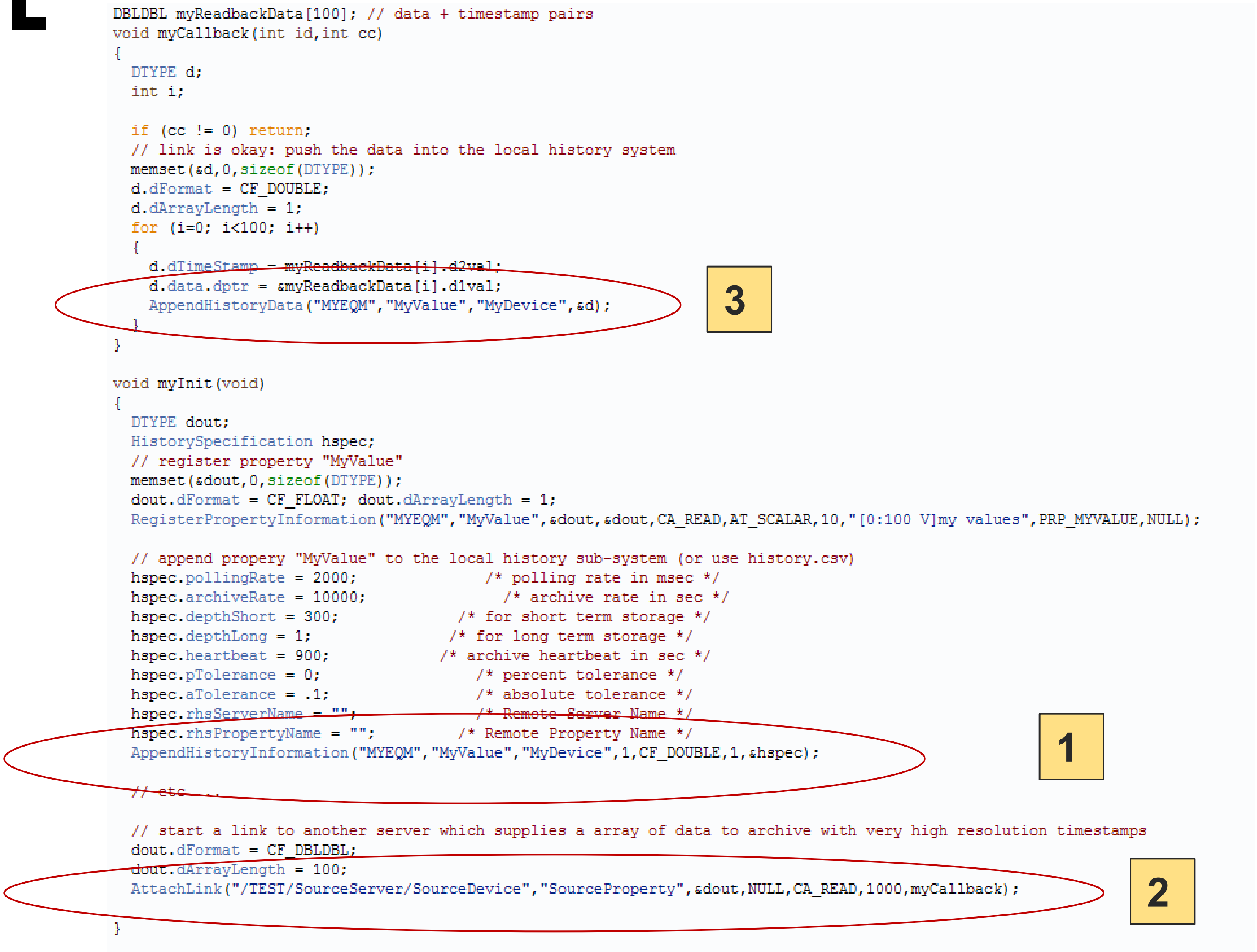
Release 4.3.2

- Embellishments and bug fixes (C and Java)
  - Globals Link now re-acquires address and multicast group when it goes down.
    - Action item from recent PETRA GLOBALS activities.
    - Note: Nonetheless it is a good idea to try to keep the IP address when moving GLOBALS from one host to another!

Release 4.3.2

- New Features (C – Lib)
  - Can now explicitly add records to the local history subsystem via API
    - AppendHistoryData()
      - Requested by MSK
      - Can obtain a set of data with very high precision timestamps with 1 call and add them piecemeal to the local history system.

Release 4.3.2 (AppendHistoryData())



```
DBLDBL myReadbackData[100]; // data + timestamp pairs
void myCallback(int id,int cc)
{
  DTYPE d;
  int i;

  if (cc != 0) return;
  // link is okay: push the data into the local history system
  memset(&d,0,sizeof(DTYPE));
  d.dFormat = CF_DOUBLE;
  d.dArrayLength = 1;
  for (i=0; i<100; i++)
  {
    d.dTimeStamp = myReadbackData[i].d2val;
    d.data.dptr = &myReadbackData[i].d1val;
    AppendHistoryData("MYEQM","MyValue","MyDevice",&d);
  }
}

void myInit(void)
{
  DTYPE dout;
  HistorySpecification hspec;
  // register property "MyValue"
  memset(&dout,0,sizeof(DTYPE));
  dout.dFormat = CF_FLOAT; dout.dArrayLength = 1;
  RegisterPropertyInformation("MYEQM","MyValue",&dout,&dout,CA_READ,AT_SCALAR,10,"[0:100 V]my values",PRP_MYVALUE,NULL);

  // append propery "MyValue" to the local history sub-system (or use history.csv)
  hspec.pollingRate = 2000;              /* polling rate in msec */
  hspec.archiveRate = 10000;               /* archive rate in sec */
  hspec.depthShort = 300;             /* for short term storage */
  hspec.depthLong = 1;                /* for long term storage */
  hspec.heartbeat = 900;             /* archive heartbeat in sec */
  hspec.pTolerance = 0;                  /* percent tolerance */
  hspec.aTolerance = .1;                 /* absolute tolerance */
  hspec.rhsServerName = "";              /* Remote Server Name */
  hspec.rhsPropertyName = "";          /* Remote Property Name */
  AppendHistoryInformation("MYEQM","MyValue","MyDevice",1,CF_DOUBLE,1,&hspec);

  // etc ...

  // start a link to another server which supplies a array of data to archive with very high resolution timestamps
  dout.dFormat = CF_DBLDBL;
  dout.dArrayLength = 100;
  AttachLink("/TEST/SourceServer/SourceDevice","SourceProperty",&dout,NULL,CA_READ,1000,myCallback);
}
```

Release 4.3.2

- New Routine (C-Lib)
  - GetRegisteredUsers(char *eqm,NAME16 *usrs, int *nusrs)
    - Requested by MSK
    - Returns those already registered users for given Equipment Module.

Release 4.3.2

- Contract Coercion News
  - Reminder: ‘*What is contract coercion?*’
    - Inefficient client requests can be coerced into more efficient, minimal load requests.
      - Via specific property registration parameters!
    - Keep unnecessary load off of the server!
    - Better to do 1 thing for many clients than many things for many clients.
    - Eschew synchronous polling !

Contract Coercion

- Bad things:
  - synchronously polling all 227 Libera BPMs one at a time at 10 Hz.
  - a timer link at 10 Hz to get DESY2 Timing data, which is already being scheduled at 6.25 Hz.
  - monitoring a list of device names or static property settings.

Data Flow Memes : 2nd Order

Contract-Coercion

- Analyze the transaction request
  - Map to an existing contract if possible
  - Anticipate future requests and renegotiate the contract with the client
    - e.g. “if he’s asking for BPM#1, then he’ll probably want BPM#2 as well”
    - Make use of Multi-Channel Arrays where possible
      - (property has registered as an **MCA**)
    - Make use of structures where possible
      - (property has registered as a STRUCT)
  - Guide synchronous and asynchronous acquisitions
    - Don’t monitor ‘static data’
    - Don’t synchronously poll monitorable data.
    - Trap ‘foolish’ update intervals

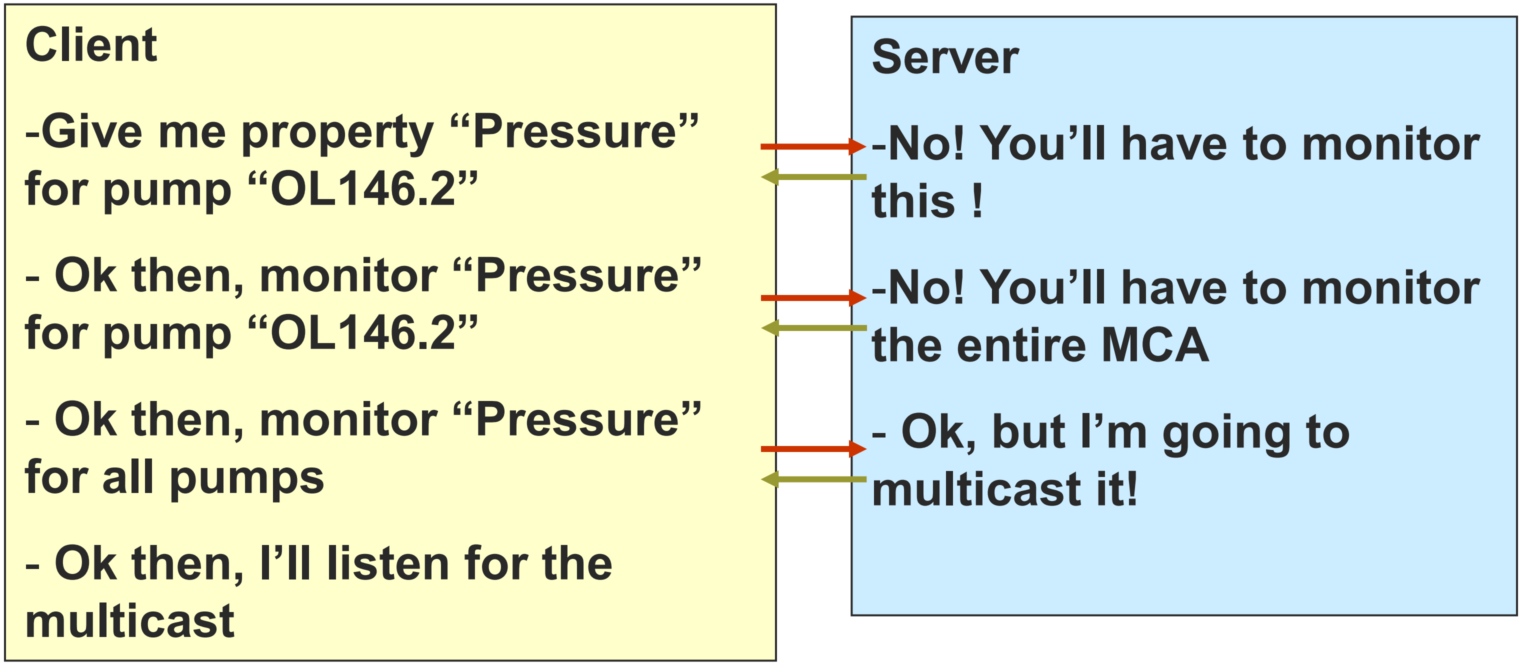
  - KISS is a distant memory

Yes, you can send structures in TINE !

A Server takes control of its Clients

Example: doing 1 thing for 1 effective client instead of 600 things for 10

A client tries to synchronously poll a single channel (e.g. a ‘get’ in a timer) …
It all happens under the hood ….



Client

-Give me property “Pressure” for pump “OL146.2”

- Ok then, monitor “Pressure” for pump “OL146.2”

- Ok then, monitor “Pressure” for all pumps

- Ok then, I’ll listen for the multicast

Server

-No! You’ll have to monitor this !

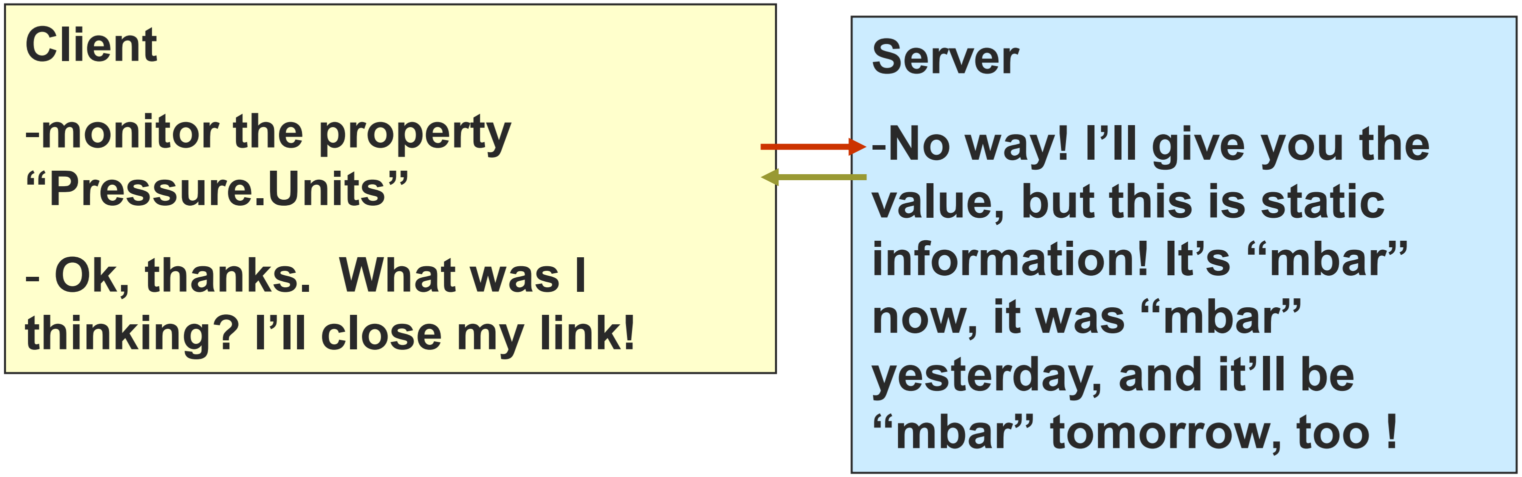
-No! You’ll have to monitor the entire MCA

- Ok, but I’m going to multicast it!

A Server takes control of its Clients

Example: Nipping superfluous polling in the bud …

A client tries to poll a static property …
It all happens under the hood ….



Client

-monitor the property “Pressure.Units”

- Ok, thanks. What was I thinking? I’ll close my link!

Server

-No way! I’ll give you the value, but this is static information! It’s “mbar” now, it was “mbar” yesterday, and it’ll be “mbar” tomorrow, too !

Contract Coercion

- What’s new:
  - trapping ‘use multicast’ and ‘asynchronous only’ messages and turning on a listener.
  - now in C and java.

Contract Coercion

- What does a server have to do to get this?
  - SetMinimumAllowedPollingInterval(1000) will stop anybody’s attempt at monitoring at a higher frequency.
  - Property Registration: apply to ‘access’
    - CA_NETWORK (to require multicast)
    - CA_NOSYNC (to require asynchronous)
    - CA_STATIC (to stop monitor attempts)
  - Property Registration: apply to ‘array type’
    - AT_CHANNEL to designate a multi-channel array
    - (or use specific registration calls)
  - etc.

Release 4.3.2

- More Local History News:
  - ready for beta-testing:
    - Can now save CF_IMAGE and CF_STRING in the local history subsystem.
      - Calls with these formats return variable data lengths !
      - Note: this was not easy!

Release 4.3.2

- Tidbits
  - Get/SetDieOnAddressInUse()
    - Default = true
    - If a server receives ‘address in use’ from the ENS, it exit(1)s with a message and log entry.
  - isDoocsServer()
    - returns ‘true’ if the target is a native doocs server.

Release 4.3.2

- *Extreme* cases
  - tineRepeater with history.csv starting 1700 links
    - they usually have < 10 links and no histories.
    - exotic requirements from PhP script for the Personnel Interlock.
    - => introduce a hash table for links in the listener logic.
      - MatLab
      - LabView
      - tineRepeater

Release 4.3.2

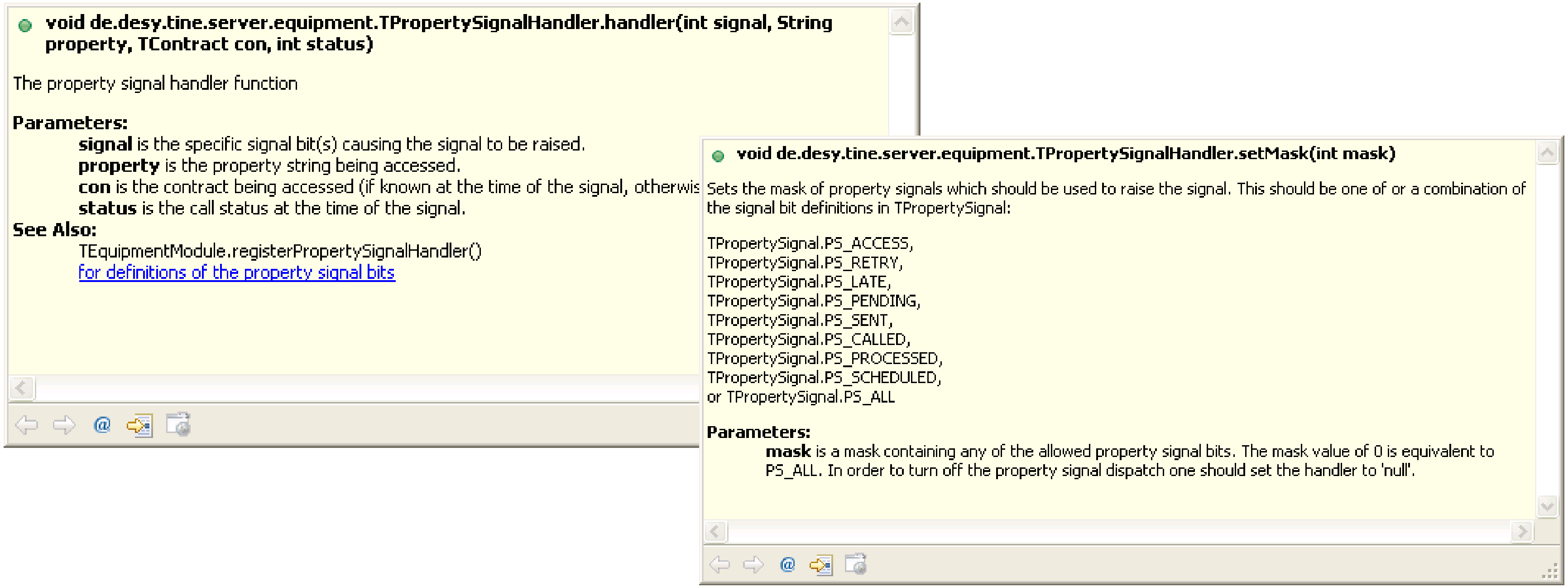
- *Extreme* cases
  - scheduling data (N x 8 Kbyte payloads) at 30 Hz from a java server
    - Thomsen Electronics (for Zeuthen)
    - Introduce property signaling a la C-Lib in java.

Release 4.3.2

- Property Signal handler

```
public class SineDeviceServer implements TLinkCallback, TPropertySignalHandler
{
  private static SineDeviceServer instance = null;
  private static SineEquipmentModule sineEqpModule;

 sineEqpModule.registerPropertySignalHandler("Sine", this);
```



void de.desy.tine.server.equipment.TPropertySignalHandler.handler(int signal, String property, TContract con, int status)

The property signal handler function

Parameters:

signal is the specific signal bit(s) causing the signal to be raised.
property is the property string being accessed.
con is the contract being accessed (if known at the time of the signal, otherwis
status is the call status at the time of the signal.

See Also:

TEquipmentModule.registerPropertySignalHandler()
for definitions of the property signal bits

void de.desy.tine.server.equipment.TPropertySignalHandler.setMask(int mask)

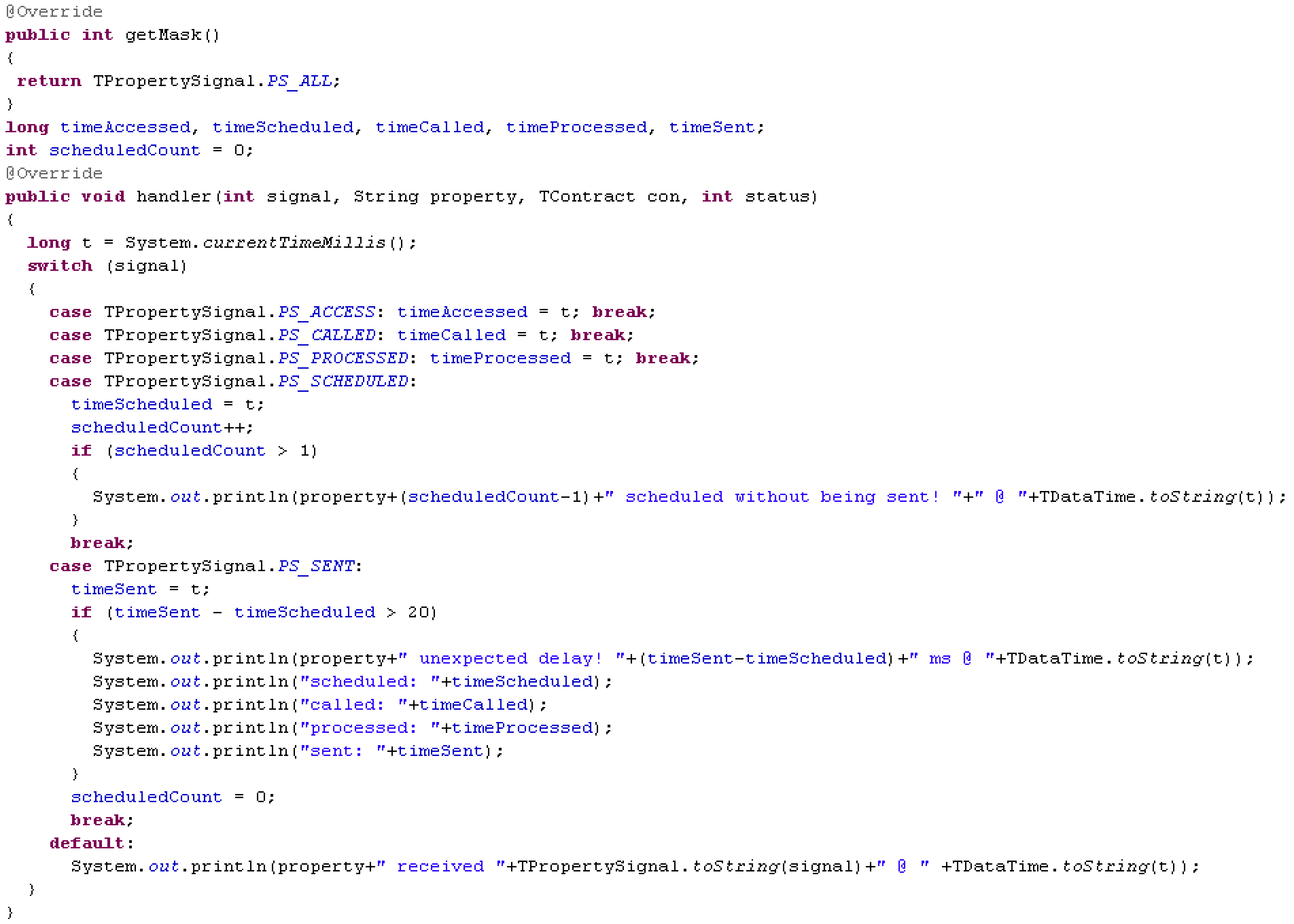
Sets the mask of property signals which should be used to raise the signal. This should be one of or a combination of the signal bit definitions in TPropertySignal:

TPropertySignal.PS_ACCESS,
TPropertySignal.PS_RETRY,
TPropertySignal.PS_LATE,
TPropertySignal.PS_PENDING,
TPropertySignal.PS_SENT,
TPropertySignal.PS_CALLED,
TPropertySignal.PS_PROCESSED,
TPropertySignal.PS_SCHEDULED,
or TPropertySignal.PS_ALL

Parameters:

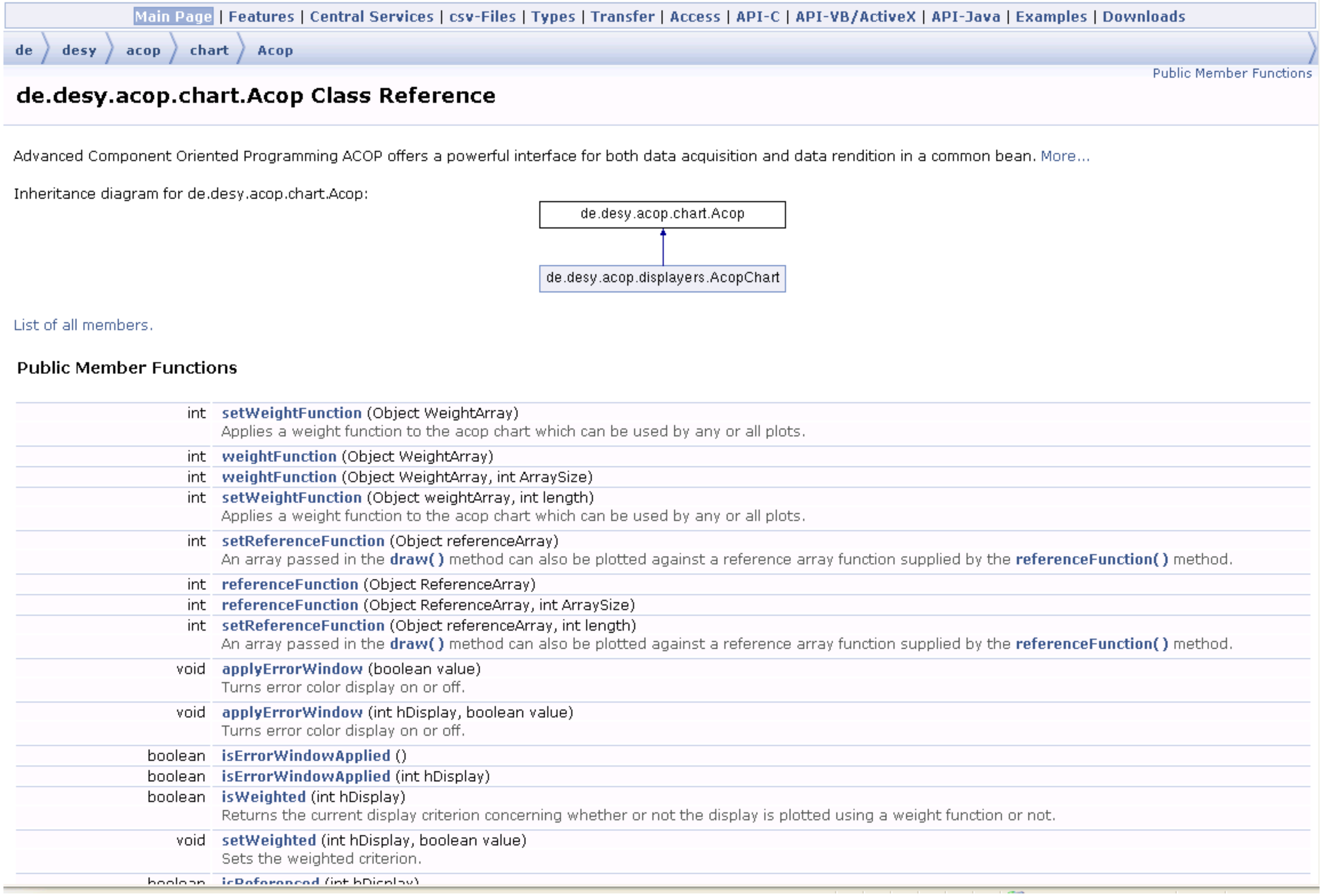
mask is a mask containing any of the allowed property signal bits. The mask value of 0 is equivalent to PS_ALL. In order to turn off the property signal dispatch one should set the handler to 'null'.

Release 4.3.2



```java
@Override
public int getMask()
{
 return TPropertySignal.PS_ALL;
}
long timeAccessed, timeScheduled, timeCalled, timeProcessed, timeSent;
int scheduledCount = 0;
@Override
public void handler(int signal, String property, TContract con, int status)
{
  long t = System.currentTimeMillis();
  switch (signal)
  {
    case TPropertySignal.PS_ACCESS: timeAccessed = t; break;
    case TPropertySignal.PS_CALLED: timeCalled = t; break;
    case TPropertySignal.PS_PROCESSED: timeProcessed = t; break;
    case TPropertySignal.PS_SCHEDULED:
      timeScheduled = t;
      scheduledCount++;
      if (scheduledCount > 1)
      {
        System.out.println(property+(scheduledCount-1)+" scheduled without being sent! "+" @ "+TDataTime.toString(t));
      }
      break;
    case TPropertySignal.PS_SENT:
      timeSent = t;
      if (timeSent - timeScheduled > 20)
      {
        System.out.println(property+" unexpected delay! "+(timeSent-timeScheduled)+" ms @ "+TDataTime.toString(t));
        System.out.println("scheduled: "+timeScheduled);
        System.out.println("called: "+timeCalled);
        System.out.println("processed: "+timeProcessed);
        System.out.println("sent: "+timeSent);
      }
      scheduledCount = 0;
      break;
    default:
      System.out.println(property+" received "+TPropertySignal.toString(signal)+" @ " +TDataTime.toString(t));
  }
}
```

Acop java doc



Main Page | Features | Central Services | csv-Files | Types | Transfer | Access | API-C | API-VB/ActiveX | API-Java | Examples | Downloads

de 〉 desy 〉 acop 〉 chart 〉 Acop

Public Member Functions

de.desy.acop.chart.Acop Class Reference

Advanced Component Oriented Programming ACOP offers a powerful interface for both data acquisition and data rendition in a common bean. More...

Inheritance diagram for de.desy.acop.chart.Acop:

de.desy.acop.chart.Acop

de.desy.acop.displayers.AcopChart

List of all members.

Public Member Functions

int	**setWeightFunction** (Object WeightArray) Applies a weight function to the acop chart which can be used by any or all plots.
int	**weightFunction** (Object WeightArray)
int	**weightFunction** (Object WeightArray, int ArraySize)
int	**setWeightFunction** (Object weightArray, int length) Applies a weight function to the acop chart which can be used by any or all plots.
int	**setReferenceFunction** (Object referenceArray) An array passed in the **draw()** method can also be plotted against a reference array function supplied by the **referenceFunction()** method.
int	**referenceFunction** (Object ReferenceArray)
int	**referenceFunction** (Object ReferenceArray, int ArraySize)
int	**setReferenceFunction** (Object referenceArray, int length) An array passed in the **draw()** method can also be plotted against a reference array function supplied by the **referenceFunction()** method.
void	**applyErrorWindow** (boolean value) Turns error color display on or off.
void	**applyErrorWindow** (int hDisplay, boolean value) Turns error color display on or off.
boolean	**isErrorWindowApplied** ()
boolean	**isErrorWindowApplied** (int hDisplay)
boolean	**isWeighted** (int hDisplay) Returns the current display criterion concerning whether or not the display is plotted using a weight function or not.
void	**setWeighted** (int hDisplay, boolean value) Sets the weighted criterion.
boolean	**isReferenced** (int hDisplay)