Tip of the Week :

- How and When to call the Scheduler

Link Transfer Modes
(a short review)

- ExecLink() or lnk.execute()
  - Synchronous CM_SINGLE link
  - Return = Transfer complete
  - Return code = Link Status
- AttachLink() or lnk.attach()
  - Asynchronous
  - Supply a callback
  - Returns immediately
  - Return = Link Id

Link Transfer Modes

- Asynchronous Link Modes (1):
  - CM_SINGLE
    - Scheduled at the server for immediate delivery : ‘no’ latency
    - Server doesn’t know whether the client said ‘execute()’ or ‘attach() + CM_SINGLE’
  - CM_DATACHANGE (CM_REFRESH)
    - polling interval tells the server when to call the scheduler.
    - data values examined following a call to the scheduler.
    - if changed -> send to caller (else not unless heartbeat)
    - can suppress notification by establishing a notification tolerance (see setNotificationTolerence())
    - possible latency

Link Transfer Modes

- Asynchronous Link Modes (2):
  - CM_TIMER (CM_POLL)
    - polling interval tells the server when to call the scheduler.
    - data sent to the caller at the polling interval.
    - can suppress notification by establishing a notification tolerance
    - possible latency
  - CM_EVENT (new to Release 4.0)
    - Server decides when to schedule the call
    - data sent to the caller following the schedule event!
    - heartbeat notification automatically suppressed.
    - ‘no’ latency
  - CM_GLOBAL
    - Receive data via Multicast in Producer-Consumer mode
    - Server end-point is effectively unknown!

Link Transfer Modes

- Modifiers:
  - CM_NETWORK
    - Send returned data to multicast group
  - CM_USE_ON_ERROR
    - Use the user supplied initial data as ‘error values’ in case of link error.
  - CM_RETRY
    - Issues a retry upon link connection error. (useful only with CM_SINGLE)
  - CM_GROUPED
    - This link is a member of a callback notification ‘Group’
  - CM_CONNECT
    - Use TCP/IP as the transport instead of UDP/IP (default)
    - n.b. TCP is a stream transport with flow control, UDP is a datagram transport with ‘best effort’.
  - CM_WAIT
    - Do not return from AttachLink() until the callback has been fired at least once.
- Not all combinations of the above make sense
- The fundamental transfer mode (previous slide) can be obtained via
  - TMode.getBaseMode(mode) (Java)
  - BASEMODE(mode) (C)

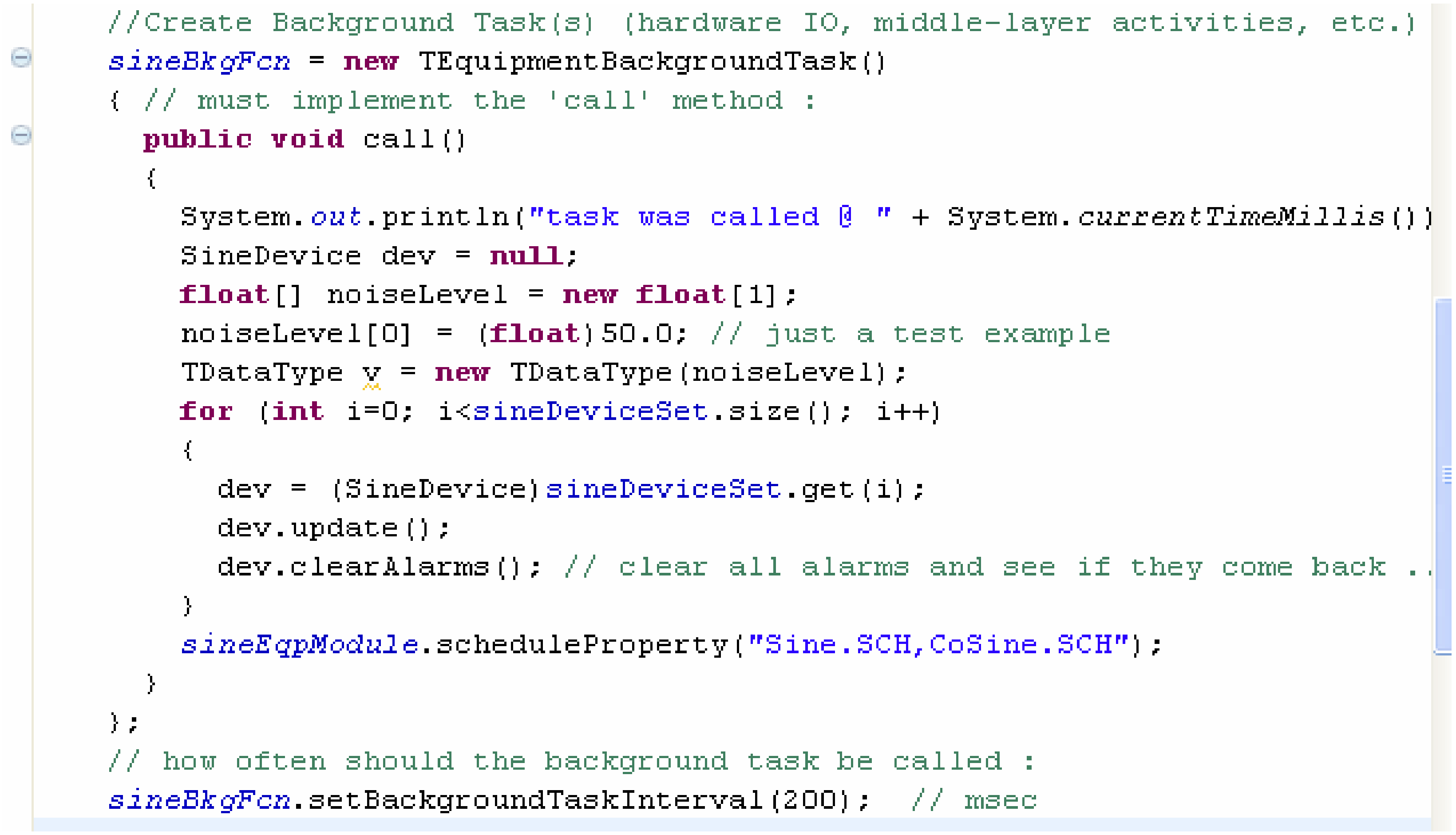
Scheduling a Transfer

- Asynchronous Transfer
  - Very efficient
  - Connection tables can bundle requests together
  - Many clients requesting the same thing latch on to the same contract
  - etc.
- But: The server is doing the scheduling!
  - You specify a polling interval with the link; the server will check the property at this interval.
  - e.g. ExecLink() will ask for a property’s value ‘now’
    - Does not mean that the data are fresh when the call returns! => could still be latency!

Scheduling a Transfer

- If latency is an issue then ‘schedule’ the property.
  - SystemScheduleProperty(eqm, prp) (C)
  - Srv.ScheduleProperty(prp) (VB)
  - TEquipmentModule.scheduleProperty(prp) (java)
  - lvTineSrvPushSingle.Vi + set the schedule to non-zero (LabView)
- Where ‘prp’ is either
  - A single registered property : “BeamCurrent”
  - A list of properties: “BeamCurrent, BeamLifetime”
- Release 4.0:
  - SystemSchedulePropertyEx(eqm, prp, scope)
  - SystemFireEvent(eqm, prp,scope) synonym for SystemSchedulePropertyEx();
  - Scope specifies
    - CA_NETWORK (all listening clients)
    - CA_HIST (the local history subsystem)
    - CA_ALARM (the local alarm subsystem)
  - Default: CA_NETWORK|CA_HIST|CA_ALARM

Java Example (Sine Server):



```
//Create Background Task(s) (hardware IO, middle-layer activities, etc.)
sineBkgFcn = new TEquipmentBackgroundTask()
{ // must implement the 'call' method :
  public void call()
  {
    System.out.println("task was called @ " + System.currentTimeMillis())
    SineDevice dev = null;
    float[] noiseLevel = new float[1];
    noiseLevel[0] = (float)50.0; // just a test example
    TDataType v = new TDataType(noiseLevel);
    for (int i=0; i<sineDeviceSet.size(); i++)
    {
      dev = (SineDevice)sineDeviceSet.get(i);
      dev.update();
      dev.clearAlarms(); // clear all alarms and see if they come back ..
    }
    sineEqpModule.scheduleProperty("Sine.SCH,CoSine.SCH");
  }
};
// how often should the background task be called :
sineBkgFcn.setBackgroundTaskInterval(200);  // msec
```

Scheduling a Transfer (side effects)

- The server developer knows which readback data belong to which property.
- Any clients listening to the scheduled Property receive an update 'immediately' regardless of their subscribed polling interval or transfer mode.
- This is a real 'event'!
- Does the client application really want this?
  - Is the Property always scheduled?
    - e.g. There's a regular external hardware trigger
  - Is the Property occasionally scheduled?
    - e.g. The readback value has crossed some interesting threshold.
  - Some category of clients might want /MHF/PE_SX_CyX/Absorber[P_RL] as a scheduled event and some not.
    - Offer two properties: P_RL and P_RL.Sched ?

Next Time :

- ???
  - Using Device Groups
  - Tagged Structures
  - Setting/Using Access Locks
  - Setting/Using data time stamps
  - Setting/Using user completion codes (returning status + data)
  - Setting/Using LinkQueue Depth vs CA_SYNCNOTIFY
  - Bit Fields (still needs to be integrated into CDI)
  - Setting/Getting Alarms
  - Setting/Getting History Data