Tip of the Week :

- Setting/Using Completion Codes

Call Completion Codes

- Some codes locally generated:
  - e.g. link_timeout, connection_timeout
- Some codes systematically generated:
  - e.g. non_existent_elem, illegal_property
- Some codes returned from the equipment module/property handler:
  - e.g. illegal_device, out_of_range, or
  - a user defined code (>= 512)

Call Completion Codes

```
switch (prpid)
{
  case PRP_SINE:
      if (access&CA_WRITE) return illegal_read_write;
      if ((cc=putValuesFromFloat(dout,sinbuf[devnr],NUM_VALUES)) != 0) return cc;
      sineInfoTable[devnr].numberCalls++;
      return 0;
  case PRP_AMPLITUDE:
      if (din->dArrayLength > 0)
      { /* input data => require write access */
        if (!(access & CA_WRITE)) return illegal_read_write;
        if ((cc=getValuesAsFloat(din,&fval,1)) != 0) return cc;
        if (fval < 1 || fval > 1000) return out_of_range;
        sineInfoTable[devnr].amplitude = fval;
      }
      if (dout->dArrayLength > 0)
      { /* prepare multichannel array */
        if (dout->dFormat == CF_FLTINT)
        {
          for (i=0; i<NUM_DEVICES && i<dout->dArrayLength; i++)
          {
            ((FLTINT *)dout->data.vptr)[i].fval = sineInfoTable[i].amplitude;
            ((FLTINT *)dout->data.vptr)[i].ival = sineInfoTable[i].numberCalls;
          }
          return 0;
        }
        for (i=0; i<NUM_DEVICES; i++)
          mcarray[i] = sineInfoTable[i].amplitude;
        if ((cc=putValuesFromFloatEx(dout,mcarray,NUM_DEVICES,devnr)) != 0) return cc;
      }
      return 0;
  case PRP_FREQUENCY:
      if (din->dArrayLength > 0)
      { /* input data => require write access */
```

Call Completion Codes

```
private int callPhase(String devName, TDataType dout, TDataType din, TAccess devAccess
{
  int cc = 0;
  SineDevice theDevice;
  if (devAccess.isWrite())
  { // CLIENT WANTS TO SET PHASE (allow single-channel write)
    double[] input = new double[1];
    if (din.getArrayLength() != 1) return TErrorList.dimension_error;
    theDevice = (SineDevice) myDeviceSet.getDevice(devName);
    if (theDevice == null) return TErrorList.illegal_equipment_number;
    if ((cc=din.getData(input)) != 0) return cc;
    theDevice.setPhase(input[0]);
  }
  if (devAccess.isRead())
  { // CLIENT WANTS TO READ (allow multi-channel read)
    int nret = dout.getArrayLength();
    if (nret < 1) return TErrorList.dimension_error;
    int ndv = myDeviceSet.getNumberOfDevices();
    int dv = myDeviceSet.getDeviceNumber(devName);
    if (dv < 0 || dv >= ndv) return TErrorList.illegal_equipment_number;
    double[] output = new double[ndv];
    for (int i=0; i<ndv; i++)
    {
      theDevice = (SineDevice) myDeviceSet.getDevice(i);
      output[i] = theDevice.getPhase();
    }
    cc = dout.putData(output,nret,dv);
  }
  return cc;
```

Call Completion Codes

- The standard error codes deliver a standard error string !
- User defined codes can also deliver a user defined string !
  - Call before returning from the eqm property handler:
    - C: SetEqmCompletion(char *eqm,char *errstr)
    - Java: setCompletionString(String errstr)
    - errstr can be 192 characters (release 4)

Call Completion Codes

- Note:
  - VB (i.e. ActiveX) and LabView both require a ‘SetCompletion()’ call to signal that the ‘event’ has been handled!

Sending a status code + Data

- You can send a non-zero return code and data back to the caller !
  - e.g. ‘has_query_function’ is used frequently in this vein!
  - e.g. eqm property handler returns
    - value_too_high + CE_SENDDATA
    - and the data (the value that was too high!)

Call Completion Codes

- =>The completion code can also carry information bits!
  - CE_REDIRECTED 0x8000
  - CE_SENDDATA   0x4000
- => caller return code should be masked against 0x0fff
  - C: GetLastLinkError()
  - Java: TErrorList.getErrorString()
  - Both of the above do this !