TINE Release 4.x.x News

(June 8, 2015: That was the month that was !)

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

Release 4.5.5

- Noteworthy Bug-fixes (C-Library)
  - The ‘attachfec’ bug …
  - The ‘tcp single byte’ bug …
- Noteworthy Bug-fixes (java)
  - Local history ‘isWithinTolerance’ + CF_NAMExx bug …
  - The CF_DEFAULT -> CF_TEXT issue …

Release 4.5.5

- The ‘attachfec’ bug :
  - Using *attachfec* to remotely attach to a multi-threaded (C-Lib) server leads to server hang-up when session is closed.
    - Introduced ver. 4.5.1 (build id 5129), 10.12.14
    - Fixed ver. 4.5.3 (build id 5134), 20.1.15

Release 4.5.5

- The TCP Single-Byte Bug :
  - If TCP Stream delivers only the initial byte of new packet *chunk* it led to an apparent data stream corruption.
    - Large payloads, very busy network
  - Fixed ver. 4.5.3 (build id 5135)

Release 4.5.5

- ‘IsWithinTolerance’ + CF_NAMExx bug:
  - Was throwing an exception.
  - Fixed 19.5.15

Release 4.5.5

- The CF_DEFAULT -> CF_TEXT issue …
  - Returned data header gives data type and size *returned*!
  - jdoocs: starts a link with CF_DEFAULT, buffer size = 128 bytes.
  - Learns that date type = CF_TEXT, but only sees ***n*** characters of a property registered to deliver ***N***.
    - e.g. receives only 10 of 80 characters.
  - jdoocs thinks that the property delivers 10 elements of type CF_TEXT !

Release 4.5.5

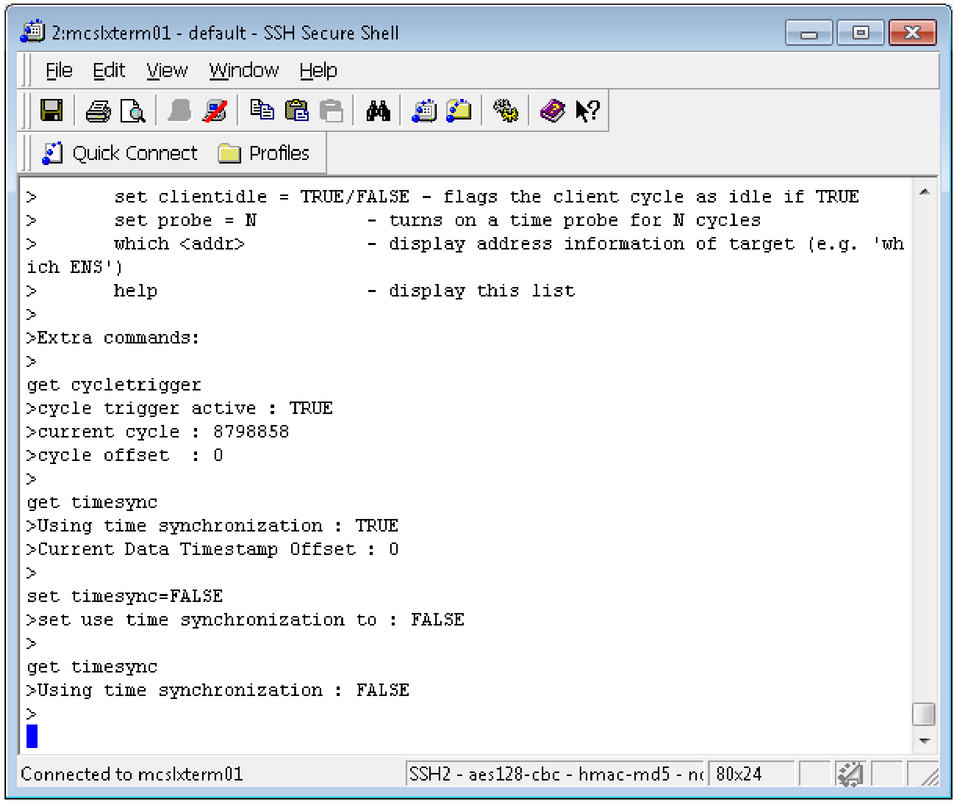
- **The CF_DEFAULT -> CF_TEXT issue …**
  - Any change in the data where > 10 characters is returned gets truncated !
    - e.g. doocs servers do just this !
    - property registered to return 80 chars only needs to return 10 so it does.
  - 1st solution: if CF_DEFAULT -> CF_TEXT did not return *buffer_too_small* then 128 was OK -> use size = 128.
    - jddd caches this learned size for future use!

Release 4.5.5

- **The CF_DEFAULT -> CF_TEXT issue …**
  - doocs servers allow requested length > registered length (great!).
  - Java Server wizard servers don’t! (oops!)
    - But they always return the registered number of characters (filled with ‘0’s).
  - 1st solution following jddd cached information and a ‘re-attach’ to a java server-wizard server lead to *dimension_error* !
- **Best strategy**: if CF_TEXT then acquire the registered property information explicitly !

Release 4.5.5

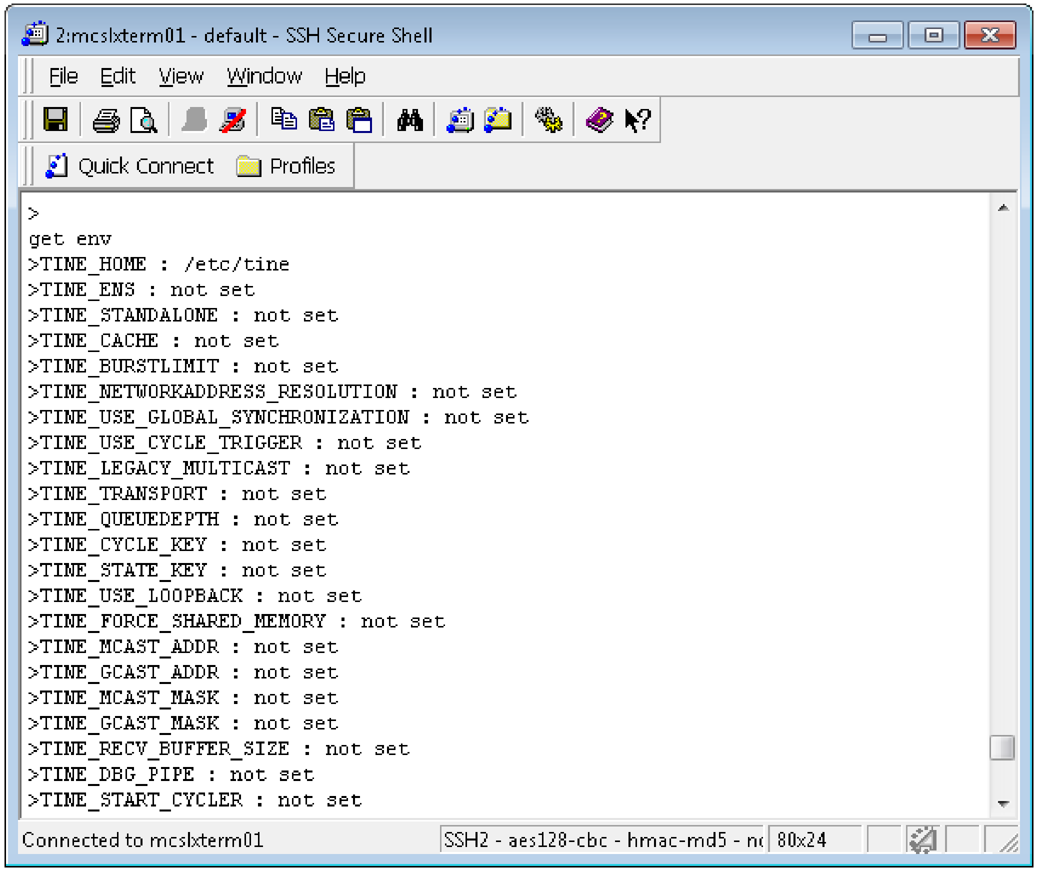
- Embellishments …
  - Can now set/get '*use cycle trigger*' at any time.
  - Can now set/get '*server time synchronization*' at any time.


2:mcslxterm01 - default - SSH Secure Shell
File Edit View Window Help
Quick Connect
Profiles

```
>       set clientidle = TRUE/FALSE - flags the client cycle as idle if TRUE
>       set probe = N          - turns on a time probe for N cycles
>       which <addr>           - display address information of target (e.g. 'wh
ich ENS')
>       help                   - display this list
>
>Extra commands:
>
get cycletrigger
>cycle trigger active : TRUE
>current cycle : 8798858
>cycle offset  : 0
>
get timesync
>Using time synchronization : TRUE
>Current Data Timestamp Offset : 0
>
set timesync=FALSE
>set use time synchronization to : FALSE
>
get timesync
>Using time synchronization : FALSE
>
```

Release 4.5.5

- Embellishments :
- Get relevant environment variable settings:


2:mcslxterm01 - default - SSH Secure Shell
File Edit View Window Help
Quick Connect
Profiles

```
>
get env
>TINE_HOME : /etc/tine
>TINE_ENS : not set
>TINE_STANDALONE : not set
>TINE_CACHE : not set
>TINE_BURSTLIMIT : not set
>TINE_NETWORKADDRESS_RESOLUTION : not set
>TINE_USE_GLOBAL_SYNCHRONIZATION : not set
>TINE_USE_CYCLE_TRIGGER : not set
>TINE_LEGACY_MULTICAST : not set
>TINE_TRANSPORT : not set
>TINE_QUEUEDEPTH : not set
>TINE_CYCLE_KEY : not set
>TINE_STATE_KEY : not set
>TINE_USE_LOOPBACK : not set
>TINE_FORCE_SHARED_MEMORY : not set
>TINE_MCAST_ADDR : not set
>TINE_GCAST_ADDR : not set
>TINE_MCAST_MASK : not set
>TINE_GCAST_MASK : not set
>TINE_RECV_BUFFER_SIZE : not set
>TINE_DBG_PIPE : not set
>TINE_START_CYCLER : not set
```

Release 4.5.5

- Java Servers:
  - Stock property ‘SRVEXIT’ now behaves as per the C-Lib Server:
    - Waits several cycles before calling System.exit().
    - Caller gets an explicit *success* when the call succeeds (instead of *link_timeout*).

Release 4.5.5

- Java Servers:
  - The issue: jddd panels *love* history displays !
    - Tend to make repeated history calls.
      - Trend chart with appends 'live data' for several seconds then 'repeats' the history call.
  - Local history files on windows:
    - NTFS horribly fragmented.
    - *Suggestion: use 'standard history files' (mkhstfiles utility) + contig.exe.*

Release 4.5.5

- Java Servers:
  - Java servers on windows: *big endian on little endian.*
    - A scan thru a large multi-channel array record involves a lot of ‘readFloat()s’ and/or byte swapping.
  - Two strikes against it (fragmentation + C-Lib file i/o is much more efficient than java for multi-channel record read-outs)

Release 4.5.5

- Java Servers:
  - CPU load goes high in a hurry when jddd is connected to a java server and opens up a history panel !
    - Watchdog was happy with ‘max cpu = 20%’ is now no longer happy!

Release 4.5.5

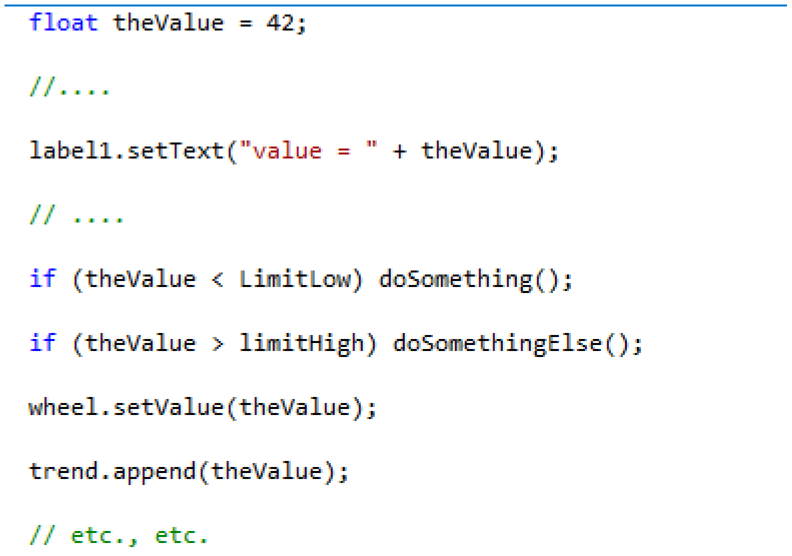
- Java Server tweaks …
  - Some problems fixed concerning ‘standard’ non-fragmented files.
  - No longer scan and read the entire record if the history of a single channel is requested !

Release 4.5.5

- Java Client Side:
  - Presenting the new data –access layer (AKA: “**The Layer**”).
  - The issue:
    - Multiple access of connection endpoints.
    - Consider rich-client programming …

Release 4.5.5 - the Layer

- Rich Client pseudo code:
  - Some value is known globally ….



```
float theValue = 42;

//....

label1.setText("value = " + theValue);

// ....

if (theValue < LimitLow) doSomething();

if (theValue > limitHigh) doSomethingElse();

wheel.setValue(theValue);

trend.append(theValue);

// etc., etc.
```

Release 4.5.5 - the Layer

- Java Client Side:
  - Consider panel-client programming …
    - There is no variable *theValue* but someone has browsed their way to
      /PETRA/Mag.Corr-NO/PKDK_NOL_86[Strom.Ist]
      on 10 different 'widgets' in a GUI designer.
    - Some of these widgets want the value once, some want to monitor on change, some want to monitor fast, some want to monitor slow, etc.

Release 4.5.5 - the Layer

- Java Client Side:
  - tine has *a layer* (there's only ever one link, client-server, to an endpoint), but it is *deep* (there is a lot of 'last-minute' checking).
  - An end-point might require extra 'learning'.
    - Is it redirected? -> if so where to?
    - Is it a single element of a multi-channel array? -> if so which one?
    - Is this one of those CF_DEFAULT things?
  - And start all the widgets off *simultaneously* each in his own thread !

Release 4.5.5 - the Layer

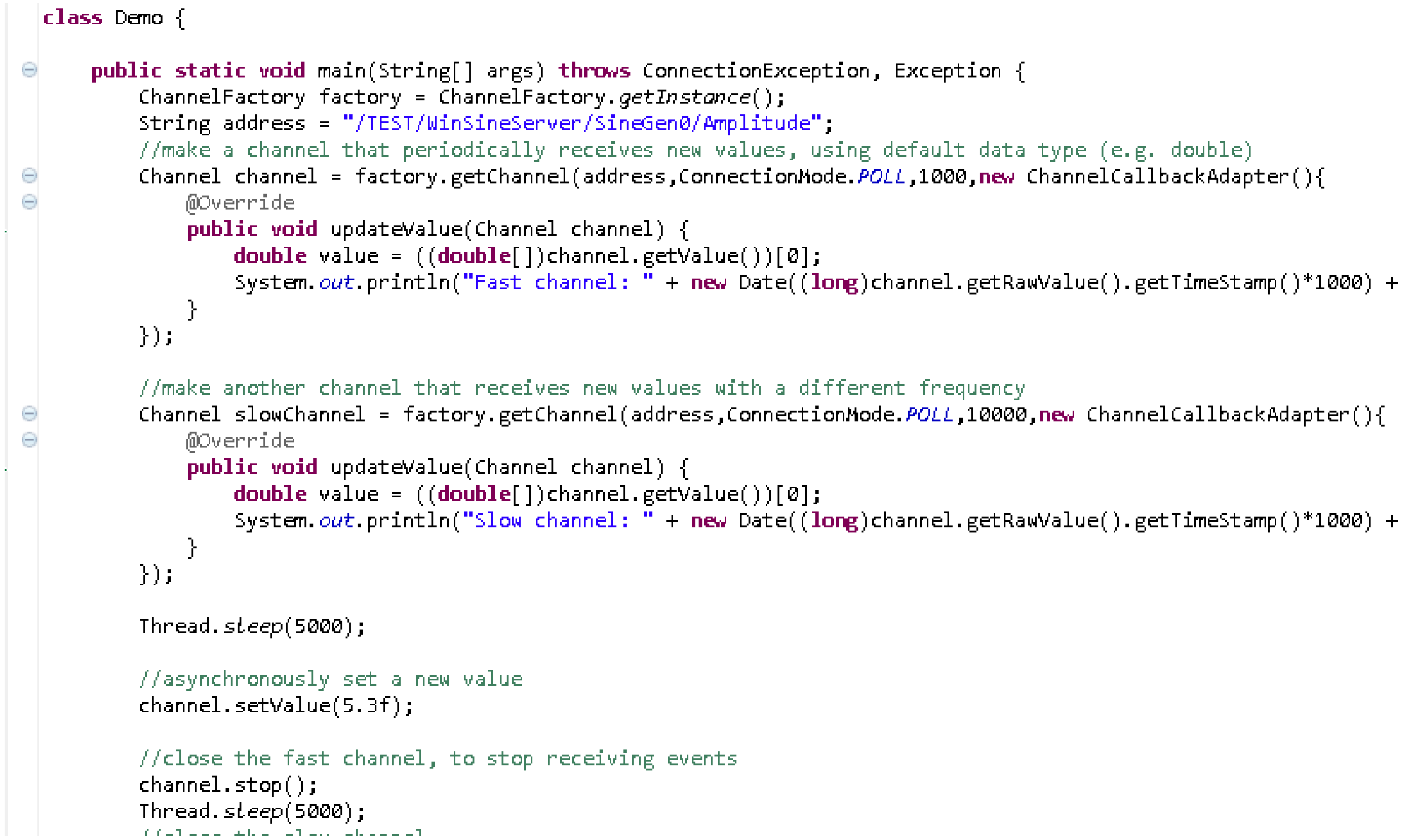
- Java Client Side:
  - Layer design:
    - Write calls feed through.
    - Read calls to static Stock Properties feed through.
    - All other read calls start by accessing the layer.
    - Everyone starts a monitor (even the single shots)
      - If single ‘gets’ stop being issued, then an idle time expires and monitor is closed.

Release 4.5.5 - the Layer

- Java Client Side:
  - Layer design:
    - Layer is *shallow*. The endpoint specifications are ‘hashed’.
      - No match -> start a new endpoint monitor.
      - Is match? attach to the monitor
      - Manage individual widget specifications in the layer.
      - Adjust timer intervals as required, etc.
      - Reflect and keep *theValue* at the client side and make all the widgets get the reflected value !

Release 4.5.5 - the Layer

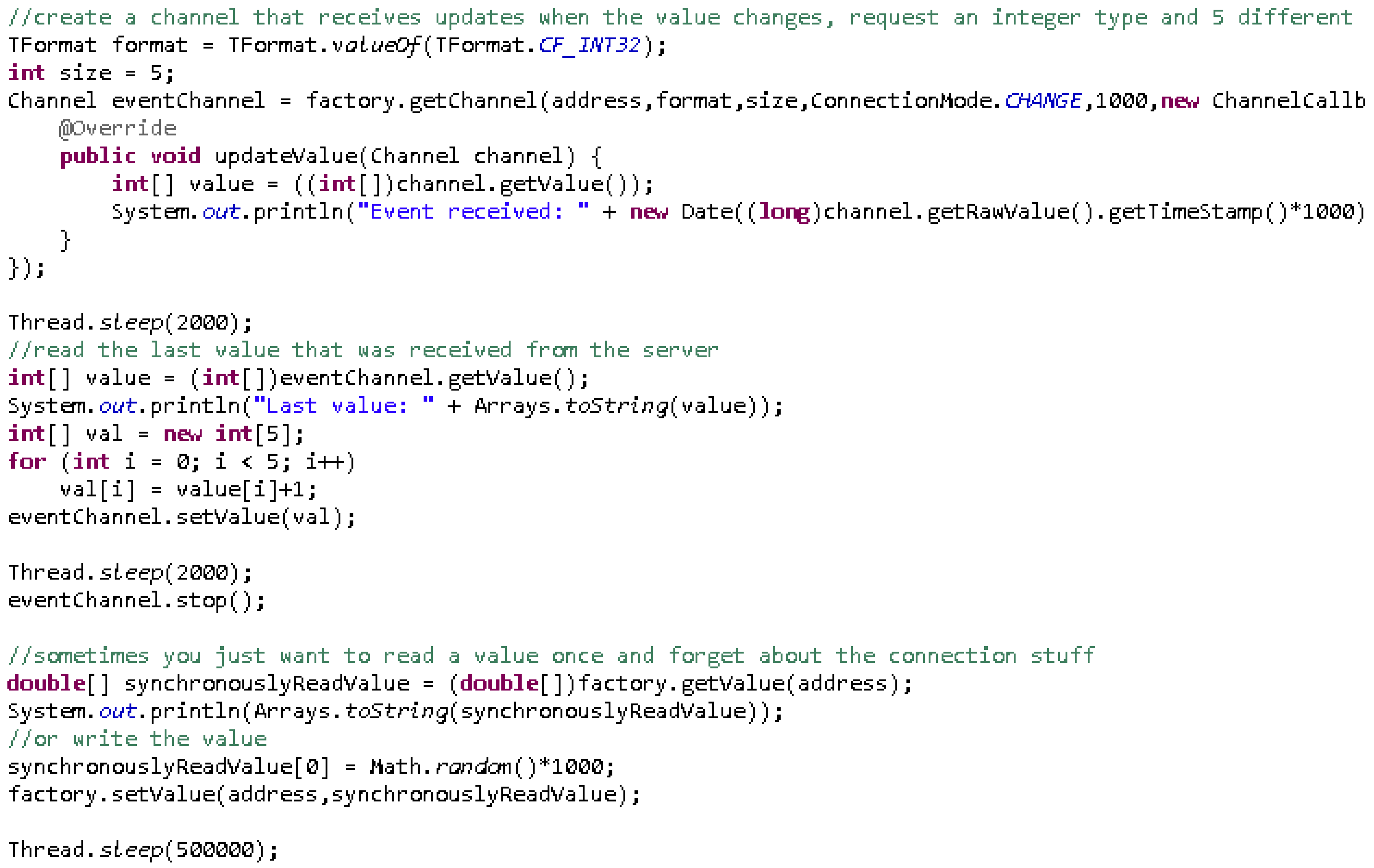
- Demo example …



```
class Demo {

    public static void main(String[] args) throws ConnectionException, Exception {
        ChannelFactory factory = ChannelFactory.getInstance();
        String address = "/TEST/WinSineServer/SineGen0/Amplitude";
        //make a channel that periodically receives new values, using default data type (e.g. double)
        Channel channel = factory.getChannel(address,ConnectionMode.POLL,1000,new ChannelCallbackAdapter(){
            @Override
            public void updateValue(Channel channel) {
                double value = ((double[])channel.getValue())[0];
                System.out.println("Fast channel: " + new Date((long)channel.getRawValue().getTimeStamp()*1000) +
            }
        });

        //make another channel that receives new values with a different frequency
        Channel slowChannel = factory.getChannel(address,ConnectionMode.POLL,10000,new ChannelCallbackAdapter(){
            @Override
            public void updateValue(Channel channel) {
                double value = ((double[])channel.getValue())[0];
                System.out.println("Slow channel: " + new Date((long)channel.getRawValue().getTimeStamp()*1000) +
            }
        });

        Thread.sleep(5000);

        //asynchronously set a new value
        channel.setValue(5.3f);

        //close the fast channel, to stop receiving events
        channel.stop();
        Thread.sleep(5000);
        //close the slow channel
```

Release 4.5.5 - the Layer

- Demo example …



```
//create a channel that receives updates when the value changes, request an integer type and 5 different
TFormat format = TFormat.valueOf(TFormat.CF_INT32);
int size = 5;
Channel eventChannel = factory.getChannel(address,format,size,ConnectionMode.CHANGE,1000,new ChannelCallb
    @Override
    public void updateValue(Channel channel) {
        int[] value = ((int[])channel.getValue());
        System.out.println("Event received: " + new Date((long)channel.getRawValue().getTimeStamp()*1000)
    }
});

Thread.sleep(2000);
//read the last value that was received from the server
int[] value = (int[])eventChannel.getValue();
System.out.println("Last value: " + Arrays.toString(value));
int[] val = new int[5];
for (int i = 0; i < 5; i++)
    val[i] = value[i]+1;
eventChannel.setValue(val);

Thread.sleep(2000);
eventChannel.stop();

//sometimes you just want to read a value once and forget about the connection stuff
double[] synchronouslyReadValue = (double[])factory.getValue(address);
System.out.println(Arrays.toString(synchronouslyReadValue));
//or write the value
synchronouslyReadValue[0] = Math.random()*1000;
factory.setValue(address,synchronouslyReadValue);

Thread.sleep(500000);
```