MT4 EA Socket Communication

Expert Advisor Communication by Sockets with the MT4 Tester

R Poster 9/2/2020

For doing analysis of the FX rate data on the MetaTrader 4 (MT4) platform, it is useful to pass results to an external program such as C++ or Python. Python has exceptional and easy to use plotting and data fitting features. However, the MetaTrader 4 Tester does not support socket communication.

I have used two workarounds for this limitation:

1. Pass data by writing cvs data files for reading by the external program. One difficulty, of course, is that the MQL4 program forces data files to be written to a specific location with a long path name related to the specific terminal. Otherwise this workaround works well.

2. Do not use the MT4 Tester but, using the OnTimer() function, read all of the historical price data and perform processing as each bar of data is read. The limitation is that the benefits of trading results are lost. After processing is complete and the information is sent over a socket, the Expert Advisor can be removed from the chart.

However, there is a way that sockets can be used with the MT4 Tester. MQL4 global data can be used to create the socket at EA initialization (OnInit()) during EA activation or recompilation. This is the non-tester mode of OnInit(). Once the socket is created and its information (context and server ID) is stored in global variables, the EA can be run in the Tester mode. The OnTester() or OnDeinit() functions can then process the results of the Tester run, read the global socket data, send processed data over the socket, close and terminate the socket, delete all global data and remove the Expert Advisor from the chart. When the EA is re-attached to the chart, the OnInit() (non tester mode) is executed and creates a new socket for the next testing activity.

The ZeroMQ library https://zeromq.org/ is an excellent and versatile tool to use for socket communication between MQL4 programs and Python programs. A library with MQH Header files and dll files has been developed for MQL4 programs by Seth and is in the MetaTrader 4 Libraries code base https://www.mql5.com/en/code/9991 .

One word of caution, when using this library, integer and double arrays are sent over to Python programs in reverse order. Also, the endian preference for MQL4 and Python data is different and must be set correctly in the Python program.

Below is an example of a program that uses global data for socket communication with the MT4 Tester.

```
//+------------------------------------------------------------------------------------------+
//|                                                   MT4Tester-Socket_Shell.mq4    |
//|                                                        Copyright 2020, R. Poster  |
//|                                                              https://www.mql5.com |
//+------------------------------------------------------------------------------------------+
//+-------------------------------------------------------------------------------------------+
//| The Purpose of this program is send plot data from MT4 Tester              |
//| over sockets to a Python program for plotting or processing                |
//| the modified ZeroMq interface software is used for socket comm            |
//| zmq_native_mqh, zmq_bind.mqh,  zmq_bind.dll, libzmq.dll                    |
//+-------------------------------------------------------------------------------------------+

#property copyright "Copyright 2019, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#include <zmq_bind.mqh>
// plot data
input  int    PortNum     =   2027;
input  int    NumPlots    =      2;
input  int    NumBins     =     25;
//
double      ydat1[],ydat2[],ydat3[],ydat4[];
double      xdat[],StepSize[4];

//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
 {
   // socket variables
   int server1,context1;
   string spn; // socket address
//
// ------------ Setup Socket Objects ------------------
   If (!IsTesting())
    {
     Print(" R/T OnInit Called ");
     // ---- socket init -----------------------
     Print("using zeromq version "+z_version_string());
     spn = "tcp://127.0.0.1:" + IntegerToString(PortNum);
     context1 = z_init(1);
     server1 = z_socket(context1,ZMQ_PUB); //server: receives queries
     if(server1==-1)
```

```
      Print(" ** Error in Socket Creation ");
 //
     if(z_bind(server1,spn)==-1) // bind for server
      {
       Print(" Socket Server Bind Error ");
       return(INIT_SUCCEEDED);
      }
     Print(" Socket Initialization Complete ");
   //  store as global to be visible to Tester
     GlobalVariableSet("context2",context1);
     GlobalVariableSet("server2",server1);
     Print(" context, sever ",GlobalVariableGet("context2"),"  ",
            GlobalVariableGet("server2"));
    }//--------------------------------------------------
   return(INIT_SUCCEEDED);
 } //--------------------------------------------------------------
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
    GlobalVariablesDeleteAll(NULL,0);
    ExpertRemove();
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//     for bar processing –v create plot data
//     store xdata, ydata for plotting by Python Matplotlib
  }
//+------------------------------------------------------------------+
//| Tester function                                                  |
//+------------------------------------------------------------------+
double OnTester()
  {
//---
   double ret=0.0;
   string xlabel,ylabel,ftitle,flegend1,flegend2,flegend3,flegend4;
   int intdata[2];
   int context,server;
   string figsubtitle="";
```

MT4 EA Socket Communication

```
//
//------------- Read Socket Data from OnInit  ----------------------------
   Print(" context, sever ",GlobalVariableGet("context2"),"  ",GlobalVariableGet("server2"));
   context = int(GlobalVariableGet("context2"));
   server =  int(GlobalVariableGet("server2"));
// -------------------- send data ----------------------------------
// ----- send integer data ------
  if(z_send_int_array(server,intdata,0,true)==-1)
   Print(" ** Send Error for Integer ");  // integer - true=> reorder array
  else
   Print(" Sent integer data ");

  Sleep(10);
// -----  send string data  ---------------
  z_send(server,ftitle);         // string
  Sleep(10);
  z_send(server,xlabel);       // string
  Sleep(10);
  z_send(server,ylabel);       // string
  Sleep(10);
  z_send(server,flegend1);   // string
  Sleep(10);
  z_send(server,flegend2);   // string
  Sleep(10);
  z_send(server,flegend3);   // string
  Sleep(10);
  z_send(server,flegend4);   // string
  Sleep(15);
// send stepsize, x and y plot data  (xstep,ystep,ylo,yhi)
  z_send_double_array(server,StepSize,0);// double - reorder array before sending
  Sleep(15);
  z_send_double_array(server,xdat,0);      // double - reorder array before sending
  Sleep(15);
  z_send_double_array(server,ydat1,0);     // double reorder array before sending
  Sleep(15);
  if(NumPlots>=2)z_send_double_array(server,ydat2,0);  // reorder array before sending
  Sleep(15);
  if(NumPlots>=3)z_send_double_array(server,ydat3,0);  // reorder array before sending
  Sleep(15);
  if(NumPlots>=4)z_send_double_array(server,ydat4,0);   // reorder array before sending

  Print("Sent Data Complete ");
  //==================================================
  z_close(server);
  z_term(context);
//---
   return(ret);
  }
//+------------------------------------------------------------------+
```