J-Data API description

Introduction :
The J-Data API enables developers to participate from the real time quotes, J-Trader and other PATS clients are providing. The API provides real time quotes (Bid, Ask,Trade and Market Depth) as well as historical data back fill for more than 2000 futures contracts and options. The API interface is developed in Visual Basic, so all variable types are Visual Basic data types.

System requirements:
The API is designed to run on all Windows versions starting with Windows 2000.
For full API functionality a working installation of a PATS client is needed on the target system. The client must be in running mode to enable real time functionality.
Quotes must be subscribed, for example by adding contracts to the hotquote window.
Real time functions only will work with real time versions of PATS clients.
Historical data functions are independent of a PATS client installation.
The API connects via port 80 with HTTP protocoll to the internet. This should be enabled in most firewalls, as it is the normal standart for web browsers.

****

Functions

Init(LicenceKey as String, UserID as String) as String
Description: Must be called, before the API will work. This should be the first call of the API. Needs a valid LicenseKey. Init generates a server log entry, which stores the UserID together with the IP address of the current computer.
Example:
Dim ReturnStr as String
ReturnStr = Init("ABC0815", "PaulSmith01")

Possible Values of "Returnstr":
"Success"
"Unknown licenceKey"
"Connection error"

Terminate()
Needs to be called before finishing the main application and before setting the reference of this API to nothing.

StartReceive(Symbol as Symbol) as String
Description: Starts to receive real time quotes for the specified symbol. Quote data will send through the events "Quote" and "MarketDepth". Calling this function several times for a symbol has no effect.
Example:
Dim TestSymbol as Symbol
Dim ReturnStr as String
TestSymbol.Exchange = "XEurex"
TestSymbol.Name = "FDAX"
TestSymbol.Date = "JUN06"
ReturnStr = StartReceive(TestSymbol)

Possible Values of "Returnstr":
"Success"
"Unknown symbol"

StopReceive(Symbol as Symbol) as String
Description: Stop receiving real time quotes for the specified symbol.
Example:
Dim TestSymbol as Symbol
Dim ReturnStr as String
TestSymbol.Exchange = "Xeurex"
TestSymbol.Name = "FDAX"
TestSymbol.Date = "JUN06"
ReturnStr = StopReceive(TestSymbol)

Possible Values of "Returnstr":
"Success"
"Unknown symbol"

DownloadHistoricalQuotes(Symbol as Symbol, FilePath as String) as String
Description: Initiates the download of historical quotes for the specified symbol. The data is saved/ appendet to the file specified in l_FilePath. The data is preformatted in 1-minute bar data in ASCII format. The date format is "MM/DD/YYYY", the time format is "HH:MM:SS"; European 24 hours format.
While this function is working, you can get the progress through the event "DownloadProgress". A data line has the following format:
Date, Time, Open, High, Low, Close, Volume
It may look like this:
"07/23/2006, 15:34:06, 5356.5, 5388, 5290.5, 5301.0, 23"
The function looks for existing historical data in l_FilePath. If the file exists, only newer historical data is downloaded. If the file does not exist, all available data is downloaded from the server (3 to 6 month).

Example:
Dim TestSymbol as Symbol
Dim ReturnStr as String
Dim FilePath as String
TestSymbol.Exchange = "Xeurex"
TestSymbol.Name = "FDAX"
TestSymbol.Date = "JUN06"
FilePath = "C:\MyQuotedata\DAXJUN06.txt"
ReturnStr = DownloadHistoricalQuotes (TestSymbol,FilePath)

Possible Values of "Returnstr":
"Success"
"Unknown symbol"
"No data"
"Unvalid path"

GetExchangeList() as String()

Description: Returns a list of all available exchanges on the historical data server. Be aware, that the client may not have all those exchanges enabled by his broker.
Example:
Dim MyExchanges() as String
MyExchanges = GetExchangeList()

GetContractList() as Contract()
Description: Returns a list of all available contracts on the historical data server. Be aware, that the client may not have all those contracts enabled by his broker.
Example:
Dim MyContracts() as New Contract
MyContracts = GetContractList()

GetSymbolDetails(Symbol As Symbol, Currency As String, StartTime As Date,
EndTime As Date, PointValue As Double, TicksPerPoint As Integer,
TickSize As Double, ExpiryDate as String,
LastTradeDate as String) As String
Description: Returns the currency, the opening time, the closing time, the point value, the number of ticks per point and the size of a tick for the specified symbol.
Example:
Dim MyContract as New Contract
Dim TestSymbol as Symbol
Dim ReturnStr as String
TestSymbol.Exchange = “XEurex"
TestSymbol.Name = “FDAX”
TestSymbol.Date = “JUN06”
ReturnStr = GetSymbolDetails(TestSymbol, MyContract.Currency, MyContract.StartTime, MyContract.EndTime, MyContract.PointValue, MyContract.TicksPerPoint, MyContract.TickSize, MyContract.ExpiryDate, MyContract.LastTradeDate)

Possible Values of “Returnstr”:
“Success”
“Unknown symbol”

****

Events

DownloadProgress(Symbol as Symbol, Date as Date)
Description: Informs you about the download progress of the named symbol. You are told, which date data was downloaded last.

MarketDepth(Symbol As Symbol, PriceType As String,
Price1 As String, Vol1 As String,
Price2 As String, Vol2 As String,
Price3 As String, Vol3 As String,
Price4 As String, Vol4 As String,
Price5 As String, Vol5 As String,
Price6 As String, Vol6 As String,
Price7 As String, Vol7 As String,
Price8 As String, Vol8 As String,

Price9 As String, Vol9 As String,
Price10 As String, Vol10 As String)

Descripton: Provides market depth for the named symbol. Price type may be “ASK” or “BID”. Be aware, that this will only work, if the client has Market Depth enabled by his broker.

Message(Message as String)
Description: Provides API system messages and warnings

Quote(Symbol As Symbol, QuoteType As Integer, Quote As Double, Volume As Long)
Description: Provides real time quotes. Quote type may be 0 (Ask), 1 (Bid) or 2 (Last Trade)

Error(Error as String)
Description: Throws an error message, if an API error has occurred.

****

Type Classes

Symbol

Exchange	as String
Name	as String
Date	as string

Contract

Exchange	as String
Name	as String
ContractDate	as String
ExpiryDate	as Date
LastTradeDate	as Date
StartTime	as Date
EndTime	as Date
PointValue	as Double
TicksPerPoint	as Integer
TickSize	as Double
Currency	as String