



## **InstaQuote Internet Protocol Link**

Version 1.5



## InstaQuote Internet Protocol Link

Version 1.5

### DISCLAIMER

THE INFORMATION CONTAINED HEREIN IS PROVIDED "AS IS" AND NO PERSON OR ENTITY ASSOCIATED WITH THE IQ IPLink MAKES ANY REPRESENTATION OR WARRANTY, EXPRESS OR IMPLIED, AS TO THE IQ IPLink (OR THE RESULTS TO BE OBTAINED BY THE USE THEREOF) OR ANY OTHER MATTER AND EACH SUCH PERSON AND ENTITY SPECIFICALLY DISCLAIMS ANY WARRANTY OF ORIGINALITY, ACCURACY, COMPLETENESS, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. SUCH PERSONS AND ENTITIES DO NOT WARRANT THAT THE IQ IPLink WILL CONFORM TO ANY DESCRIPTION THEREOF OR BE FREE OF ERRORS. THE ENTIRE RISK OF ANY USE OF THE IQ IPLink IS ASSUMED BY THE USER.

NO PERSON OR ENTITY ASSOCIATED WITH THE IQ IPLink SHALL HAVE ANY LIABILITY FOR DAMAGES OF ANY KIND ARISING IN ANY MANNER OUT OF OR IN CONNECTION WITH ANY USER'S USE OF (OR ANY INABILITY TO USE) THE IQ IPLink, WHETHER DIRECT, INDIRECT, INCIDENTAL, SPECIAL OR CONSEQUENTIAL (INCLUDING, WITHOUT LIMITATION, LOSS OF DATA, LOSS OF USE, CLAIMS OF THIRD PARTIES OR LOST PROFITS OR REVENUES OR OTHER ECONOMIC LOSS), WHETHER IN TORT (INCLUDING NEGLIGENCE AND STRICT LIABILITY), CONTRACT OR OTHERWISE, WHETHER OR NOT ANY SUCH PERSON OR ENTITY HAS BEEN ADVISED OF, OR OTHERWISE MIGHT HAVE ANTICIPATED THE POSSIBILITY OF, SUCH DAMAGES.

No proprietary or ownership interest of any kind is granted with respect to the IQ IPLink (or any rights therein).  
Copyright 2002 Direct Access Financial Corporation, all rights reserved



## Revision History

Rev 5: Tag 132 = TimeLast was added to the Required Tag list under Message ID 501.

Rev 6: Tag 126 = MOO was added to the Optional Tag list under Message ID 102.  
Tag 127 = MOC was added to the Optional Tag list under Message ID 102.  
Tag 128 = ISLD Invisible was added to the Optional Tag list under Message ID 102.  
Tag 129 = ISLD Show was added to the Optional Tag list under Message ID 102.  
Tag 130 = FOK was added to the Optional Tag list under Message ID 102.  
Tag 131 = NYSE Auto-Ex (NYSE Direct+) was added to the Optional Tag list under Message ID 102.  
Tag 132 = Pegging Orders have been added.  
Tag 153 = Removed.  
Tag 100 = Updated to be included in all messages  
Tag 101 = Updated to be included in all messages  
Rev 7: Tag 135 = Discretionary Field added  
Clarified the Login procedures

## Overview

IQ IPLink is a lightweight point-to-point protocol built on top of TCP/IP sockets that allows a set of sequenced messages to be flow between a server and a client.

## Carrier Requirements

IQ IPLink does not demand a single type of carrier (e.g., it will work with leased lines, frame relay, Internet, etc.), nor a single security protocol. It leaves many of these decisions to the individual firms that are using it.

## Data Packet Structure

The IQ IPLink client and server communicate by passing a series of logical messages back and forth. Each message has a variable length payload and is terminated with an ASCII 3 decimal character.

Each message starts with 4 binary 0's (4 bytes of '00000000') followed by a tagged list of fields.



# InstaQuote - API Link

Tags are delimited using ASCII character 01 and the complete message is terminated with ASCII character 03

Tag 1 (Message Length) and Tag 2 (Message ID) must always be part of the message.

Each message starts as follows:

00001=length|2=MsgID|.... the rest of the message...

## **IQ IP data packet structure**

Note that the IQ IP logical packets do not necessarily map directly to physical packets on the underlying network socket; they may be broken apart or aggregated by the TCP/IP stack.

## **Protocol Flow**

A connection begins with the client opening a TCP/IP socket to the server and sending a Login Request message. If the login request is valid, the server responds with a Login Accepted message and begins sending data. The connection continues until the TCP/IP socket is broken.

## **Heartbeats**

IQ IPLink uses logical heartbeat packets to quickly detect link failures. The server must send a Server Heartbeat packet anytime more than 1 second has past since the server last sent any data. This ensures that the client will receive data on a regular basis. If the client does not receive anything (neither data nor heartbeats) for an extended period of time, it can assume that the link is down and attempt to reconnect using a new TCP/IP socket.

Similarly, once logged in, the client must send a Client Heartbeat packet anytime more than 1 second has past since the client last sent anything. If the server doesn't receive anything from the client for an extended period of time (typically 15 seconds), it can close the existing socket and listen for a new connection.

## **Data Types**

Character data fields are standard ASCII bytes. Numeric fields use ASCII digits and are padded on the left with spaces.



# InstaQuote - API Link

In the following examples, we use the '|' character to denote the separators between the tags. The terminating character (ASCII 3) is not shown.

## Heartbeat from server:

00001=0048|2=1|99=2|999=Timeout Interval is 20 secs

MsgLength	MsgID	RetCode	Text
1=0048	2=1	99=2	999=Timeout Interval is 20 secs

## Sample Login:

00001=57|2=101|3=1|100=DEMOAPI|101=USER|102=PASSWRD|103=1004

MsgLength	MsgID	VerNo	Domain	UserID	Password	Account
1=57	2=101	3=1	100=DEMOAPI	101=USER	102=PASSWRD	103=1004

## LoginResponse:

00001=0018|2=500|99=0

MsgLength	MsgID	RetCode
1=0018	2=500	99=0

## Example of a Buy:

00001=0107|2=102|103=1004|104=0|105=DELL|106=12|108=1|109=1|110=100|111=22.450000|112=ARCA|100=DEMOAPI|101=USER

MsgLength	MsgID	Account	AccType	Symbol	Exchange	Side	Type	Shares	Price	Route
1=0107	2=102	103=1004	104=0	105=DELL	106=12	108=1	109=1	110=100	111=22.450000	112=ARCA

Domain	UserID
100=DEMOAPI	101=USER



# InstaQuote - API Link

## Order Detail:

00001=0193|2=501|97=1|98=2|100=DEMOAPI|101=USER|103=1004|104=0|105=DELL|106=12|108=1|109=1|110=100|111=12.100000|112=ARCA|120=20020227-1001|121=2|122=100|123=25.720000|124=100|125=DEMO|998=|999=|

MsgLength	MsgID	Historic	BlockSeq	Domain	UserID	Account	AccType
1=0193	2=501	97=1	98=2	100=DEMOAPI	101=USER	103=1004	104=0

Symbol	Exchange	Side	Type	Shares	Price	Route	OrderNo
105=DELL	106=12	108=1	109=1	110=100	111=12.100000	112=ARCA	120=20020227-1001

LastStatus	LastShares	LastPrice	ExeShares	ECNid	Memo	Text
121=2	122=100	123=25.720000	124=100	125=DEMO	998=	999=

## Position Details:

00001=0113|2=502|98=2|100=DEMOAPI|103=1004|104=0|105=DELL|106=12|140=1|141=222|142=0.000000|142=0.000000|145=|146=12

MsgLength	MsgID	BlockSeq	Domain	Account	AccType	Symbol	Exchange	Position	PrevShares	DayShares
1=0113	2=502	98=2	100=DEMOAPI	103=1004	104=0	105=DELL	106=12	140=1	141=222	142=0.000000

OptionsRoot Symbol	OptionRoot Exchange
145=	146=2

## Example of an Account status:

00001=0121|2=503|100=DEMOAPI|103=1004|150=1000000.000000|151=250000.000000|152=1000000.000000|153=0.000000|154=250000.000000|

HeartBeat	MsgID	Domain	Account	BPBuyStock	BPBuyOptions	BPSHORTStock
1=0121	2=503	100=DEMOAPI	103=1004	150=1000000.000000	151=250000.000000	152=1000000.000000

BPSellCoverOpt	BPSellNakedOpt
153=0.000000	154=250000



## TAG List:

<p><b>The first 4 bytes of each Message must be binary '0' followed by a Free format string that is driven off a "tag=" list separated the 0x01 and terminated by 0x03. ALL DATA UNLESS OTHERWISE SPECIFIED MUST BE SENT IN UPPERCASE !!!!! This includes data concerning USER, DOMAIN, PASSWORD, and SYMBOL.</b></p>		
<b>Tag List</b>		
<b>1 = MsgLength</b>	This field must always be present as the first field. The message length is defined as the length, in bytes of the message, excluding the 4 header 0's but including all the separation and termination fields. Failure to send the length, or if the terminator is not found at MsgLength characters from Msgstart, will result in the connection being terminated without any warning.	
<b>2 = MsgID</b>	This field must always be present as the second field, This field will identify the nature of the data/instructions contained in the message. Only the data/instructions of the MsgID that is the second field will be processed. Failure to include this tag as a second message will result in the message not being processed.	
<b>3 = VerNo</b>	This field is used to identify the version of the messages that will be sent. Used in 101 Login Msg. Please note, that the current software version number is 1. This number is not the same as the client version number. Changes to the version number will be detailed in the revision history.	
<b>97 = Historic</b>	<p>This flag indicates whether or not the message is part for the Historic Data sent during the Post-Login Data Transfer. This was added to make programming easier.</p> <p>Message msg=501 with tag 99=99 mark the official end of the Post-Login Data Transfer. The Flag (97) will be set to <b>1</b> to indicate historic data and omitted in all other cases.</p>	
<b>98 = BlockSeq</b>	This tag indicates the sequence in a block transfer. It is used to show what to expect during the transfer of a sequence of data.	
	1 & 3 Do not carry data.	
	To avoid unnecessary messages 0 can be used to send just a single item. A typical empty list would be 1 & 3 that would indicate no Data found, but then reset the local Database and complete the transfer. The main implementation of this tag is just after the login when the server sends all the historic data to the client.	
	0	Single Item (Data)
	1	Block Start (No Data)
	2	Block Item (Data)
	3	Block End (No Data)
<b>99 = RetCode</b>	0	This tag indicates a success



# InstaQuote - API Link

	1>= (Greater than or Equal to)	Error code in some msg's
	Ex: 500 login response. Also the RetCode the server needs from your heartbeat response.	
<b>100 = Domain</b>	This tag must always be present in all Messages. This field will be checked against the logged-in user on that port if the field does not match the connection will be terminated without any warning.	
<b>101 = UserID</b>	This tag must always be present in all Messages. This field will be checked against the logged-in user on that port if the field does not match the connection will be terminated without any warning.	
<b>102 = Password</b>	This tag is required in the login Message. It is strongly recommended that this field is not populated in any further messages to minimize the possible unlawful detection of passwords. Remember, you are not just exposing your client, but the data integrity of every one on the system.	
<b>103 = Account</b>	This tag must always be present in all messages indicated. This field will be checked against the logged-in user at that port if the field does not match the connection will be terminated without any warning.	
<b>104 = AccType</b>	0 = Margin 1 = Cash	
<b>105 = Symbol</b>	Symbol Name	
<b>106 = Exchange</b>	Exchange Code	
	11	Bulletin Board
	12	Option
	13	NYSE
	14	AMEX
	15	NASDAQ
<b>107 = Region</b>	Exchange Region Code K=Bulletin Board	
<b>108 = Side</b>	1	Buy
	2	Sell
	3	Sell Short
<b>109 = Type</b>	1	Market
	2	Limit
<b>110 = Shares</b>	Number of shares ordered	
<b>111 = Price</b>	Price for Order. *Please note that the price must be set to 0 for Market Orders.	
<b>112 = Route</b>	Executing ECN ex. ISLD, ARCA, ADP, SOES, SNET or "Default" for auto route.....	
<b>113 = StopType</b>	1	The number 1 denotes that the system will Stop out at a Fixed StopAmount.
	2	The number 2 denotes that the system will Stop out at a Trailing StopAmount.
<b>114 = StopAmount</b>	Fixed Stop Price for <b>StopType=1</b> Trailing Stop Amount for <b>StopType=2</b>	
<b>115 = Gtc</b>	Set to 1 for Good Till Cancel Order	
<b>116 = Ext</b>	Set to 1 for Extended Hour Trading	





# InstaQuote - API Link

<b>117 = PrefMMID</b>	Preferred MarketMaker ID identifies the Market Maker or ECN that a route may preference.	
<b>120 = OrderNo</b>	InstaQuote OrderNo. This is unique per order and domain.	
<b>121 = LastStatus</b>	This tag indicates the status of the order at this detail Line.	
	-2	Denotes that the order has been Received from the Client.
	-1	Denotes that the order has been Placed at the ECN.
	0	Denotes that the order has been Confirmed by ECN.
	1	Denotes that the order has been Part Filled.
	2	Denotes that the order has been Filled.
	3	Denotes that the order has Expired/Done for the day (Remaining shares Canceled).
	4	Denotes that the Remaining shares have been Canceled.
	8	Denotes that the order has been Rejected (Remaining shares Canceled).
<b>122 = LastShares</b>	Denotes shares Executed on this detail line.	
<b>123 = LastPrice</b>	Denotes the Price of the shares Executed on this detail line.	
<b>124 = ExeShares</b>	Denotes the Total Shares Executed for this order up to and including the detail line.	
<b>125 = ECNid</b>	This is the ECN that the Route sent the order to. 99% of the time it will be the same as Route, Demos go to the DEMOECN.	
<b>126 = Market On Open</b>	Order Type is Market on Open	
<b>127 = Market On Close</b>	Order Type is Market On Close	
<b>128 = ISLD-INV</b>	Order Type is Island Invisible	
<b>129 = ISLD-Show</b>	Order Type is Island Show	
<b>130 = FOK</b>	Order Type is Fill or Kill	
<b>131 = NYSE Auto-Ex</b>	Order Type is NYSE Auto-Ex or Direct+	
<b>132 = Pegged Orders</b>	1	Peg Market
	2	Peg Best
	3	Peg Last
	4	Peg Mid
	5	Peg Primary
<b>134 = AutoSell</b>	Optional Tag – If AutoSell is not desired, leave Tag 134 out.	
	1	To enable Simultaneous selling of a Hedge's underlying security set to "1".
<b>140 = Position</b>	Indicates what type of position is held.	
	0	Short
	1	Long
<b>141 = PrevShares</b>	Details the number of Shares in Current Position that comes from overnight positions.	
<b>142 = DayShares</b>	Details the number of Shares in Current Position that comes from transactions done today.	
<b>143 = PrevCost</b>	Details the Cost of Shares in Current Position that comes from overnight positions.	
<b>144 = DayCost</b>	Details the Cost of Shares in Current Position that comes from transactions done today.	
<b>145 = OptionRootSymbol</b>	Root Symbol Name - if Symbol (105) is an option.	



<b>146 = OptionRootExchange</b>	Exchange Code - if Symbol (105) is an option, See Tag (106) for Values.
<b>147 = Region</b>	Exchange Region Code - if Symbol (105) is an option, See Tag (106) for Values.
<b>150 = BPBuyStock</b>	Buypower Available for Buying Stock.
<b>151 = BPBuyOptions</b>	Buypower Available for Buying Option.
<b>152 = BPShortStock</b>	Buypower Available for Shorting Stocks.
<b>154 = BPSellNakedOpt</b>	Buypower Available for Selling Naked Options.
<b>998 = Memo Sent or Received</b>	This Field can be used to add a client memo to orders when placed. This Memo Field will be sent back to the client on the Received confirmation so the memo field can be matched up with an Instaquote OrderNo.
<b>999 = Text</b>	All text comments, including reject reasons, and system messages. This field should not be parsed since the format is not fixed or documented. Its main use is to provide a readable version of message's that can be passed on to the front-end user.

## Message ID List:

Required Tags list assumes that Tags 1=MsgLength and 2=MsgID are required for all messages and that Tags 100=Domain, and 101=UserID are required for all messages 100-499		
<b>Message ID List</b>		
<b>1 = HeartBeat</b>	This message must be sent by the server +/- every second (1), the client must send the exact message back to the server within the time limit stated in the text msg or the server will disconnect the client.	
	<b>Required Tags:</b>	
	99	RetCode (Server generated Sequence Number)
	999	Text (States what the current server timeout is set to)
	<b>Optional Tags:</b>	
	None	None
<b>101 = Login</b>	Logon request message. It must always be the first message sent, any failure to do so will result in the connection being terminated without any warning. PLEASE read msgID=1 regarding duplicated connections.	
	<b>Required Tags:</b>	
	3	VerNo
	102	Password
	103	Account
	<b>Optional Tags:</b>	
	None	None
<b>102 = PlaceOrder</b>	This Message will place new orders into the market	
	<b>Required Tags:</b>	



# InstaQuote - API Link

	103	Account
	104	Account Type
	105	Symbol
	106	Exchange
	108	Side
	109	Type
	110	Shares
	111	Price
	112	Route
	100	Domain
	101	User
	<b>Optional Tags:</b>	
	107	Region
	113	Stop Type Must Have Tag = 114 Set
	114	Stop Amount Must be present with Tag 113
	115	GTC
	116	Ext
	117	PrefMMID
	126	Order Type is Market on Open
	127	Order Type is Market On Close
	128	Order Type is Island Invisible
	129	Order Type is Island Show
	130	Order Type is Fill or Kill
	131	Order Type is NYSE Auto-Ex or Direct+
	998	Memo
	999	Text
<b>103 = CancelOrder</b>	This message will send a Cancel Request to the market. Remember, this is only a request until your cancellation request has been confirmed. Until then consider this order live. As always, it is very important to remember that executions can still be reported after a cancellation request.	
	<b>Required Tags:</b>	
	103	Account
	105	Symbol
	106	Exchange
	120	OrderNo
	<b>Optional Tags:</b>	
	107	Region



# InstaQuote - API Link

<b>500 = LoginResponse</b>	Login response Message. This message will notify the client of a successful or failed login. It is important not to consider a client logged in until all of the post login data has been received.	
	0 means Login expected, but 99 will signal when the Historical Data has been transmitted and the client can continue sending trades and other messages.	
	<b>Required Tags:</b>	
	99	RetCode (0=success; 1=failed;2=duplicate login;99=Post-Login Data Transfer Done)
<b>501 = OrderDetails</b>	<b>Optional Tags:</b>	
	999	Text
	This tag carries all of the details regarding an order. Ex. reviewed/placed/executed. After a successful login, all of the historic Order Details are also sent to the client as a data block utilizing tag=98.	
	<b>Required Tags:</b>	
	98	BlockSeq
	100	Domain
	101	UserID
	103	Account
	104	Account Type
	105	Symbol
	106	Exchange
	108	Side
	109	Type
	110	Shares
	111	Price
	112	Route
	120	OrderNo
	121	LastStatus
	122	LastShares
	123	LastPrice
	124	ExeShares
	125	ECNid
	132	Last Time
	<b>Optional Tags:</b>	
	97	Historic
	107	Region

# InstaQuote - API Link

	113	Stop Type Must Have Tag = 114 Set
	114	Stop Amount Must be present with Tag=113
	115	GTC
	116	Ext
	117	PrefMMID
	998	Memo
	999	Text
<b>502 = PositionDetails</b>	This message carries all of the details regarding the current open positions in the account. After a successful login, all of the current open positions are sent to the client as a data block utilizing tag=98.	
	Updates are sent as trading day goes on.	
	<b>Required Tags:</b>	
	98	BlockSeq
	100	Domain
	103	Account
	104	Account Type
	105	Symbol
	106	Exchange
	140	Position
	141	PrevShares
	142	DayShares
	143	PrevCost
	144	DayCost
	145	OptionRootSymbol
	146	OptionRootExchange
	147	OptionRootRegion
	<b>Optional Tags:</b>	
	97	Historic
	107	Region
<b>503 = AccStat</b>	This message carries all of the details regarding an account's Buying Power. This Message will be sent at login and again every time one the fields updates during the session.	
	<b>Required Tags:</b>	
	100	Domain
	103	Account
	150	BPBuyStock
	151	BPBuyOptions
	152	BPSHORTStock
	153	BPSellCoverOpt



# InstaQuote - API Link

	154	BPSellNakedOpt
	<b>Optional Tags:</b>	
	97	Historic

## IP Link Demo

### What it is

The IpLinkDemo application was developed as a sample of the features of IQ's IPLink.

As such, it is not a complete trading application, and exists solely to demonstrate one way of implementing the IQ IPLink specification.

### How it was written

The application was written in C++ using the MFC framework.

Programmers who are not familiar with the MFC application framework may be confused by all the code in the application, but most of that code is generated by the MFC Application Wizard, and does not have much significance in the actual application's data flow.

### Getting Started

The first step is to unzip the sample using WinZip or a compatible product. Take note that the Use Folder Names option must be checked so that the RES directory is created correctly.

Once that is done, open the file *IpLinkDemo.dsp* in Microsoft Visual Studio, using the Open Project option in the File Menu.

Once that is done, you build the project as per normal.

When you run the application, a login screen is displayed, followed by a Trade dialog that allows you to place trades to the API server. All output is then displayed in the document window.

All the different windows in the sample application interact with the *IPLink* module, which contains all the code to create IQ IPLink messages, as well as the code to convert the incoming messages into a manageable form.

Incoming messages consist of a string of tags. The *IPLink module* contains code, which extracts all the tags and builds a CString Array which has the values at the appropriate positions in the array.

So, when a message is received, it is processed and tag 1's text will be in the String Array's index 1, Tag 2 is at CString[2], etc.



# InstaQuote - API Link

The *TranslateMsg* function in IPLINK.cpp is where this code resides.

## The Modules:

Here is a list of all the modules, with a short description of each:

Module	Description
ChildFrm.cpp	MFC generated. No demo code here.
ChildFrm.h	MFC generated. No demo code here.
IpLinkDemo.cpp	Main MFC module. Added code in InitInstance function to display IQ windows
IpLinkDemo.h	Main header for IpLinkDemo.cpp
IpLinkDemo.rc	Resource file – Contains the dialogs
IpLinkDemo.reg	Reg file used by MFC and the registry
IpLinkDemo.dsp	WorkSpace – open this file in Developer Studio
StdAfx.cpp	MFC application main module - empty
StdAfx.h	MFC application main header
resource.h	Resource Header.
Res	Directory containing MFC generated resources
IPLink.cpp	This is where most of the communication functions are located
IPLink.h	This is the header for IpLink.cpp
IpLinkDemoDoc.cpp	Document manager – has function to add text to the output window
IpLinkDemoDoc.h	Header file for IpLinkDemoDoc.cpp
IpLinkDemoView.cpp	View Manager for the document – Displays all messages to and from server
IpLinkDemoView.h	Header file for IpLinkDemoView.cpp
IPLSock.cpp	Small wrapper around MFC's Socket interface.
IPLSock.h	Header for the socket wrapper.
IPL_Util.cpp	Some utility functions here to convert Tag numbers to descriptive text
IPL_Util.h	Header file for the utility functions
logdump.cpp	This contains a function to dump debugging information to a file.
MainFrm.cpp	MFC generated. Does contain functions to display our trade and details windows.
MainFrm.h	Header for MainFrm.cpp
setupdlg.cpp	Dialog code to get the user, password, domain, and IP address of the server
setupdlg.h	Header for setupdlg.cpp
TradeDlg.cpp	Trade Demo example – dialog which gathers data for an execution
TradeDlg.h	Header for TradeDlg.cpp
TradeListView.cpp	This is the trade details window manager – displays trade status
TradeListView.h	Header file for TradeListView.cpp



# InstaQuote - API Link

## Current Restrictions

None Known

