OMEGA ATS

Omega Alternative Trading System

Omega SOUP Interface Specification Guide

OMEGA ALTERNATIVE TRADING SYSTEM

Interface and Protocol Specifications

(Version 1.0.20)

Table of Contents

REVISION HISTORY **3**

PURPOSE **4**

SOUPTCP 2.00 SPECIFICATION **5**

INTRODUCTION 5

SOUPTCP LOGICAL PACKETS 5

Protocol Flow *6*

Heartbeats *6*

End of Session Marker *6*

Data Types *7*

SOUPTCP PACKET TYPES 7

Debug Packet *7*

Logical Packets Sent by a SoupTCP Server *7*

Logical Packets Sent by the SoupTCP Client *8*

ADDENDUM A ..... ERROR! BOOKMARK NOT DEFINED.

FIX AND THE TRANSPORT LAYER: TCP/IP ..... **ERROR! BOOKMARK NOT DEFINED.**

What is TCP/IP? ..... ***Error! Bookmark not defined.***

TCP/IP Port Connection ..... ***Error! Bookmark not defined.***

CONNECTIVITY ..... **ERROR! BOOKMARK NOT DEFINED.**

Internet (Various Providers) ..... ***Error! Bookmark not defined.***

BT Radianz ..... ***Error! Bookmark not defined.***

Transaction Network Services, Inc. TNS ..... ***Error! Bookmark not defined.***

Savvis ..... ***Error! Bookmark not defined.***

BANDWIDTH REQUIREMENTS ..... **ERROR! BOOKMARK NOT DEFINED.**

Revision History

Date	Version	Description	Author
August 13, 2007	1.0.8	First Public Release	Norman Bates
September 28, 2007	1.0.9	Added FIX Tags required for Compliance and Reporting	NB
October 12, 2007	1.0.10	Drop SymbolSfx(65), now use Symbol dot-suffix notation	NB
October 18, 2007	1.0.11	Added Execution Report ExecRefID(19) to the document	NB
November 19, 2007	1.0.12	Removed ITCH2.0a section, modified ITCH3.0 to 10-char symbols and added UMIRInventoryMatchFlag to the Execution Report	NB
March 22, 2008	1.0.13	Added OnBehalfOf and DeliverTo Tags to Header, Executing Exchange and MaxFloor to support iceberg and hidden orders.	NB
August 15, 2008	1.0.14	Added Program Trades (6755) to New Order – Single, Program Trades (6755) and Wash Trades (6777) to Execution Reports Changed UMIRInventoryClientMatch (6752) in Execution Reports to tag 6776	Raymond Tung
September 18, 2008 October 15, 2008	1.0.15	In ITCH, added Message Type 'B' for Trade Busts. Added SecondaryOrderID (198) to Execution Reports Added Protection (6820) and ProtectionPriceImprovement (6821) to New Order – Single Changed Anonymous from Tag 7012 to Tag 6761 in New Order – Single and added tag to Execution Report	RT
November 3, 2008	1.0.16	Added tag SecurityExchange (207) to New Order - Single Added tags SecurityIDSource (22), SecurityID (48), and SecurityExchange (207) to Execution Report	RT
January 8, 2009	1.0.17	Updated ITCH specification with: Starting sequence # in Login Request Packet changed from '0' to '1'. Changed the Execbroker length from 4 characters to 3. Changed the Execbroker value from 'OMG' to a numerical value. Added a Reserved field with offset 40 and length 1 after the Execbroker field.	RT
April 20, 2009	1.0.18	Correction - For FIX tag OrdStatus (39), added value "9" for suspended Added tag UMIRBypass (6791) to New Order – Single Added tag UMIRBypass (6791) to Execution Report	RT
January 12, 2010	1.0.20	Created separate spec for SOUP transport	Greg King

Purpose

The purpose of this SOUP Interface Protocol Specification is that of a Guide for participants and to act as the Outline for test and verification steps, which must be performed for Certification of Interfaces to the Omega ATS System.

Subscribers are encouraged to review additional literature and the full text of the OMEGA ATS Subscriber Manual located at website address: http://www.omegaats.com/subscribe.php.

SoupTCP 2.00 Specification

Introduction

The information in this section is specific to the Omega Alternative Trading System. The entire SoupTCP 2.00 specification is available from NASDAQ at:

http://www.nasdaqtrader.com/trader/mds/nasdaqfeeds/souptcp.pdf

SoupTCP is a lightweight point-to-point protocol, built on top of TCP/IP sockets that allow delivery of a set of sequenced messages from a server to a client in real-time. SoupTCP guarantees that the client receives each message generated by the server in sequence, even across underlying TCP/IP socket connection failures.

SoupTCP clients can send messages to the server. These messages are not sequenced and may be lost in the case of a TCP/IP socket failure.

SoupTCP is ideal for systems where a server needs to deliver a logical stream of sequenced messages to a client in real-time but does not require the same level of guarantees for client generated messages either because the data stream is unidirectional or because the server application generates higher-level sequenced acknowledgments for any important client-generated messages.

SoupTCP is designed to be used in conjunction with higher lever protocols that specify the contents of the messages that SoupTCP messages deliver. The SoupTCP protocol layer is opaque to the higher-level messages, except that the messages carried by SoupTCP may not include the ASCII linefeed character and must be at least 1 byte long.

SoupTCP also includes a simple scheme that allows the server to authenticate the client on login.

SoupTCP Logical Packets

The SoupTCP client and server communicate by exchanging a series of logical packets.

Each SoupTCP logical packet has:

- a single byte header which indicates the packet type
- a variable length payload
- a terminating linefeed character (ASCII 10 decimal, 0x0A hex)

Packet Type	Variable length payload	Terminating line feed character

The SoupTCP 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.

The SoupTCP protocol does not define a maximum payload length.

The payload may not contain the line feed character.

Protocol Flow

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

Each Sequenced Data Packet carries a single higher-level protocol message.

Sequenced Data Packets do not contain an explicit sequence number; instead both client and server compute the sequence number locally by counting messages as they go.

The sequence number of the first sequenced message in each session is always 1.

Typically, when initially logging into a server the client will set the Requested Sequence Number field to 1 and leave the Requested Session field blank in the Login Request Packet. The client will then inspect the Login Accepted Packet to determine the currently active session. Starting at 1, the client begins incrementing its local sequence number each time a Sequenced Data Packet is received. If the TCP/IP connection is ever broken, the client can then re-log into the server indicating the current session and its next expected sequence number. By doing this, the client is guaranteed to always receive every sequenced message in order, despite TCP/IP connection failures.

SoupTCP also permits the client to send messages to the server using Unsequenced Data Packets at any time after the Login Accepted Packet is received. These messages may be lost during TCP/IP socket connection failures.

Heartbeats

SoupTCP 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.

End of Session Marker

The server indicates that the current session has terminated by sending a Sequenced Data Packet containing a zero length message in the payload. This indicates that there will be no more messages contained in this session.

The client will have to reconnect and re-login with a new Session ID to begin receiving messages for the next available session.

Data Types

Character data fields are standard ASCII bytes.

Numeric fields use ASCII digits and are padded on the left with spaces.

SoupTCP Packet Types

Debug Packet

Name	Offset	Length	Value	Comments
Packet Type	0	1	'+'	Debug Packet
Text	1	Variable	Alphanumeric	Free form human readable text
Terminating Linefeed	Text length +1	1	Linefeed Character	ACSII 10 decimal, 0x0A hex

Logical Packets Sent by a SoupTCP Server

LOGIN ACCEPTED PACKET

The SoupTCP server sends a Login Accepted Packet in response to receiving a valid Login Request from the client. This packet will always be the first non-debug packet sent by the server after a successful login request.

Name	Offset	Length	Value	Comments
Packet Type	0	1	'A'	Login Accept Packet
Session	1	10	Alphanumeric	Session ID of the session that is now logged into
Sequence Number	11	10	Numeric	Session ID of the session that is now logged into
Terminating Linefeed	21	1	Linefeed Character	ACSII 10 decimal, 0x0A hex

LOGIN REJECTED PACKET

The SoupTCP server sends this packet in response to an invalid Login Request Packet from the client. The server closes the socket connection after sending the Login Reject Packet. The Login Rejected Packet will be the only non-debug packet sent by the server in the case of an unsuccessful login attempt.

Name	Offset	Length	Value	Comments
Packet Type	0	1	'J'	Login Rejected Packet
Reject Reason Code	1	1	Alpha	"A" Not Authorized – invalid user/password "S" Sessioninvlaid or not available
Terminating Linefeed	2	1	Linefeed Character	ACSII 10 decimal, 0x0A hex

SEQUENCED DATA PACKET

The Sequenced Data Packets act as an envelope to carry the actual sequenced data messages that are transferred from the server to the client. Each Sequenced Data Packet carries one message from the higher-lever protocol. The sequence number of each message is implied; the initial sequence number of the first Sequenced Data Packet for a given TCP/IP connection is specified in the Login Accepted Packet and the sequence number increments by 1 for each Sequenced Data Packet transmitted.

Since SoupTCP logical packets are carried via TCP/IP sockets, the only way logical packets can be lost is in the event of a TCP/IP socket connection failure. In this case, the client can reconnect to the server and request the next expect sequence number and pick up where it left off.

Name	Offset	Length	Value	Comments
Packet Type	0	1	'S'	Sequenced Data Packet
Message	1	Variable	Alphanumeric	Data defined by a higher protocol without embedded linefeeds. Zero length indicates no more messages available
Terminating Linefeed	Payload length + 1	1	Linefeed Character	ACSII 10 decimal, 0x0A hex

SERVER HEARTBEAT PACKET

The server should send a Server Heartbeat Packet anytime more than 1 second passes where no data has been sent to the client. The client can then assume that the link is lost if it does not receive anything for an extended period of time.

Name	Offset	Length	Value	Comments
Packet Type	0	1	'H'	Server Heartbeat Packet
Terminating Linefeed	1	1	Linefeed Character	ACSII 10 decimal, 0x0A hex

Logical Packets Sent by the SoupTCP Client

LOGIN REQUEST PACKET

The SoupTCP client must send a Login Request Packet immediately upon establishing a new TCP/IP socket connection to the server.

Client and server must have mutually agreed upon the username and password fields. They provide simple authentication to prevent a client from inadvertently connecting to the wrong server.

Both Username and Password are case-insensitive and should be padded on the right with spaces.

The server can terminate an incoming TCP/IP socket if it does not receive a Login Request Packet within a reasonable period of time (typically 30 seconds).

Name	Offset	Length	Value	Comments
Packet Type	0	1	'L'	Login Request Packet
Username	1	6	Alphanumeric	Username
Password	7	10	Alphanumeric	Password
Requested Session	17	10	Alphanumeric	Specifies session to log onto. All blanks to log onto the currently active session
Requested Sequence Number	27	10	Numeric	Specifies the next sequence number the client wants to receive upon connection or '1' to start from the first message. (Sequence number is padded with spaces to the left)
Terminating Linefeed	37	1	Linefeed Character	ACSII 10 decimal, 0x0A hex

UNSEQUENCED DATA PACKET

The Unsequenced Data Packets act as an envelope to carry the actual data messages that are transferred from the client to the server. These messages are not sequenced and may be lost in the event of a socket failure. The higher-level protocol must be able to handle these lost messages in the case of a TCP/IP socket connection failure.

Name	Offset	Length	Value	Comments
Packet Type	0	1	'U'	Unsequenced Data Packet
Message	1	Variable	Alphanumeric	Data defined by a higher protocol without embedded linefeeds
Terminating Linefeed	Payload length + 1	1	Linefeed Character	ACSII 10 decimal, 0x0A hex

CLIENT HEARTBEAT PACKET

The client should send a Client Heartbeat Packet anytime more than 1 second passes where no data has been sent to the server. The server can then assume that the link is lost if it does not receive anything for an extended period of time.

Name	Offset	Length	Value	Comments
Packet Type	0	1	'R'	Client Heartbeat Packet
Terminating Linefeed	1	1	Linefeed Character	ACSII 10 **decimal**, 0x0A hex

LOGOUT REQUEST PACKET

The client may send a Logout Request Packet to request the connection be terminated. Upon receiving a Logout Request Packet, the server will immediately terminate the connection and close the associated TCP/IP socket.

Name	Offset	Length	Value	Comments
Packet Type	0	1	'O'	Logout Request Packet
Terminating	1	1	Linefeed	ACSII 10 **decimal**, 0x0A hex

Linefeed			Character	