SoupTCP

Version 2.00

1. Overview

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.

1.1 SoupTCP Logical Packets

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

Each SoupTCP logical packet has:

A. a single byte header which indicates the packet type

B. a variable length payload

C. a terminating linefeed character (ASCII 10 decimal, 0x0A hex).

Packet Type	Variable-length payload	Terminating line feed character

SoupTCP Logical Packet Structure

Notes:

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.

1.2 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.

If the client indicates a next expected sequence number that is in excess of the current highest sequenced message the server will not deliver messages until it passes the client's expected sequence. Clients in this scenario should log back in with an expected sequence of 1.

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.

1.3 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.

1.4 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.

1.5 Data Types

Character data fields are standard ASCII bytes.

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

2. SoupTCP Packet Types

2.1 Debug Packet

A debug packet can be sent by either side of a SoupTCP connection at anytime. Debug packets are intended to provide human readable text that may aid in debugging problems. Debug Packets should be ignored by both client and server application software.

Debug Packet

Name	Offset	Len	Value	Notes
Packet Type	0	1	"+"	Debug Packet
Text	1	Variable	Alpha-numeric	Free form human readable text.

Terminating Linefeed	Text Len+1	1	Linefeed Character	ASCII 10 decimal, 0x0A hex.

2.2 Logical Packets Sent by a SoupTCP Server

2.2.1 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.

Login Accepted Packet

Name	Offset	Len	Value	Notes
Packet Type	0	1	"A"	Login Accepted Packet
Session	1	10	Alpha-numeric	The session ID of the session that is now logged into. Left padded with spaces.
Sequence Number	11	10	Numeric	The sequence number of the next Sequenced Message to be sent. Left padded with spaces.
Terminating Linefeed	21	1	Linefeed Character	ASCII 10 decimal, 0x0A hex.

2.2.2 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.

Login Reject Packet

Name	Offset	Len	Value	Notes

Packet Type	0	1	"J"	Login Rejected Packet
Reject Reason Code	1	1	Alpha	See Login Reject Codes below.
Terminating Linefeed	2	1	Linefeed Character	ASCII 10 decimal, 0x0A hex.

Login Reject Codes

Code	Explanation
"A"	Not Authorized. There was an invalid username and password combination in the Login Request Message.
"S"	Session not available. The Requested Session in the Login Request Packet was either invalid or not available.

2.2.3 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.

Sequenced Data Packet

Name	Offset	Len	Value	Notes
Packet Type	0	1	"S"	Sequenced Data Packed
Message	1	Variable	Alpha-numeric	Defined by a higher-level protocol, but must not contain any embedded

				linefeeds. A Message with zero length is a special End of Session marker that indicated that there are no more messages available in this session.
Terminating Linefeed	Payload Len+1	1	Linefeed Character	ASCII 10 decimal, 0x0A hex.

2.2.4 Sever 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.

Server Heartbeat Packet

Name	Offset	Len	Value	Notes
Packet Type	0	1	"H"	Server Heartbeat Packet
Terminating Linefeed	1	1	Linefeed Character	ASCII 10 decimal, 0x0A hex.

2.3 Logical Packets Sent by the SoupTCP Client

2.3.1 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).

Login Request Packet

Name	Offset	Len	Value	Notes
Packet Type	0	1	"L"	Login Request Packet
Username	1	6	Alphanumeric	Username
Password	7	10	Alphanumeric	Password
Requested Session	17	10	Alphanumeric	Specifies the session the client would like to log into, or all blanks to log into the currently active session.
Requested Sequence Number	27	10	Numeric	Specifies the next sequence number the client wants to receive upon connection, or 0 to start receiving the most recently generated message.
Terminating Linefeed	37	1	Linefeed Character	ASCII 10 decimal, 0x0A hex.

2.3.2 Unsequenced Data Packets

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.

Unsequenced Data Packet

Name	Offset	Len	Value	Notes
Packet Type	0	1	"U"	Unsequenced Data Packed
Message	1	Variable	Alpha-numeric	Defined by a higher-level protocol, but must not contain any embedded linefeeds.
Terminating Linefeed	Payload Len+1	1	Linefeed Character	ASCII 10 decimal, 0x0A hex.

2.3.3 Client Heartbeat Packets

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.

Client Heartbeat Packet

Name	Offset	Len	Value	Notes
Packet Type	0	1	"R"	Client Heartbeat Packet
Terminating Linefeed	1	1	Linefeed Character	ASCII 10 decimal, 0x0A hex.

2.3.4 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.

Logout Request Packet

Name	Offset	Len	Value	Notes
Packet Type	0	1	"O"	Logout Request Packet
Terminating Linefeed	1	1	Linefeed Character	ASCII 10 decimal, 0x0A hex.

3. Support

Any questions about the SoupTCP specification should be emailed to devsupport@nasdaq.com.

4. Current Restrictions

None known.

5. Revision History

Version	Date	Revision History
1.0	09/01/1999	Initial Distribution
2.0	10/29/2001	Added Heart beats in both directions to remove the dependence on TCP/IP keepalives to detect link failures. Server and client are now both guaranteed to send something (either data or heartbeat) at least once per second. This way, if you don't hear anything from the socket for several seconds, you can assume the socket is dead and close it. Added Debug Messages because they are handy for debugging problems. An example is to have a server send a Debug Message upon accepting a connection identifying the name of the machine. This way, someone can TELNET into a server and immedeately verify that they have the right host. Added "envelopes" around both the outbound Sequenced Messages and the inbound Unsequenced Message to differentiate them from the heartbeats.
2.0	03/20/2014	Added language to deal with scenario where user submits a next expected sequence number that is in excess of the current highest sequence message on the server.