INDUSTRIAL CONTROL COMMUNICATIONS, INC.

Generic Socket Client Driver Manual

October 30, 2014

© 2014 Industrial Control Communications, Inc.

TABLE OF CONTENTS

1 Generic Socket Client

1.1 Overview

The Generic Socket client driver can be used to access ASCII and raw data on a variety of TCP and UDP servers, such as barcode scanners, weigh scales and serial-to-Ethernet converters. The driver uses a connection object element to target the end device. The connection object defines a connection to a specific endpoint (IP address and port number) and defines the behavior of the connection.

Some notes of interest are:

- Both TCP and UDP connections are supported. TCP is connection-oriented and guarantees the delivery and order of packets. UDP is faster than TCP, but does not guarantee the delivery or order of packets.
- The driver supports up to 10 total connection objects (TCP and/or UDP).
- Both automatic and manual triggering is supported.

1.2 Connection Object Behavior

For TCP connection objects, a persistent TCP socket connection is immediately established with the server upon boot-up. In some cases, the mere act of establishing this connection acts as notification to the server that it may begin transmitting unsolicited data at some device-specific rate. In other cases, the server may require that the client transmit a specific Starting Sequence in order to evoke a corresponding data transmission (similar to a typical "request/response" paradigm).

For UDP connection objects (which are by nature connectionless), a Starting Sequence can be used to act as notification to the server that it may transmit data. If the server does not require a Starting Sequence, then it must continuously transmit unsolicited data at some device-specific rate, as the client driver will never transmit anything to the server, or establish any sort of connection (the generic socket client driver is, in the purest sense, in "listen only" mode).

Appropriate connection object and Scan Rate configuration must take into account the conditions required by the server to produce data, as well as the server's rate of data production (if unsolicited).

1.3 Client Settings

Scan Rate

This is the time in milliseconds that each connection object will wait between processing actions. A processing action includes transmitting a starting sequence (if enabled) and processing any received data. The scan rate is a useful feature to reduce overall network utilization, or for certain devices or infrastructure components (such as radio modems) that may not be capable of sustaining the maximum packet rates that the driver is capable of producing. The start time for this delay is taken with respect to the moment at which the driver completes the previous processing action (due to either successful reception of data or a timeout). If no additional time is required, setting this field to 0 instructs the driver to initiate its next processing action as soon as possible.

1.4 Connection Object Settings

With the single exception of the “Source Port” field available in UDP Connection Objects, both UDP and TCP Connection Object configuration is identical. To simplify the documentation, this section will therefore jointly summarize the settings for both types.

Name

This 32-character (max) field is strictly for user reference: it is not used at any time by the driver.

IP Address

Defines the IP address of the server device to be targeted by the connection object. All TCP connection object *IP Address* and *Destination Port* setting combinations must be unique.

Destination Port

Defines the destination port (1…65535) of the server device, which must be monitoring this port for incoming requests.

- For TCP connection objects, this port defines both the target of the persistent socket connection as well as the *Transmit Starting Sequence* (if enabled).
- For UDP connection objects, this field is only available if a *Transmit Starting Sequence* is in use (if a *Transmit Starting Sequence* is not used, then the driver never sends anything to the server device.)

Source Port

This field is available only for UDP Connection Objects. Defines the UDP port (1…65535) on which the driver will transmit requests to the server device (if a *Transmit Starting Sequence* is in use), as well as monitor for incoming data from the server device.

Transmit Starting Sequence

Check this checkbox to send an optional starting sequence to the server. Among other possible uses, a starting sequence may be necessary to evoke the transmission of response data from the server. Check the server device’s documentation to determine whether or not a starting sequence is required.

Starting Sequence

This field is available only when the Transmit Starting Sequence checkbox is enabled. Enter the *Starting Sequence* required by the server as a string of hexadecimal values, two characters at a time. For example, entering “7E” in this field will result in the single hexadecimal byte 0x7E being transmitted in the starting sequence. If the server requires an ASCII-based sequence, then enter the appropriate ASCII codes (e.g. entering “4D36” results in the ASCII characters “M6” being sent by the driver). Up to 100 hexadecimal value codes can be entered in this field.

Receive Termination Mode

Select the appropriate receive termination mode. Once the termination mode is satisfied, the received data is considered complete and will be processed by the driver. The available selections are:

Termination Sequence	The server sends a specific termination sequence.
Message Timeout	After initial detection of incoming data, the driver will wait a specified amount of time (the *Receive Delay* setting). All further incoming data will be processed until the receive delay time has elapsed.
Character/Packet Gap Delay	After initial detection of incoming data, all further incoming data will be processed until a reception gap occurs that meets or exceeds the *Receive Delay* setting.
Fixed Size	The driver processes all data until a specified amount of data is received.

Receive Termination Sequence

This field is available only when Receive Termination Mode is set to "Termination Sequence". Enter the termination sequence generated by the server as a string of hexadecimal values, two characters at a time. For example, enter "7E" in this field if the server transmits the single hexadecimal byte 0x7E as its termination sequence. If the server generates an ASCII-based sequence, then enter the appropriate ASCII codes (e.g. enter "4D36" if the server transmits the two ASCII characters "M6" as its termination sequence). One or two hexadecimal value codes can be entered in this field.

Strip Receive Termination Sequence

This field is available only when Receive Termination Mode is set to "Termination Sequence". Check this checkbox to strip the termination sequence prior to storing data in the database. If unchecked, the termination sequence will be stored in the database along with the packet data.

Receive Delay

This field is available only when Receive Termination Mode is set to "Message Timeout" or "Character/Packet Gap Delay".

- In "Message Timeout" mode, this is the amount of time during which all received data will be processed after initial detection of incoming data. This timer is started once when received data is initially detected, and expiration of this timer terminates reception.
- In "Character/Packet Gap Delay" mode, this is the length of the reception gap time. This timer is first started when received data is initially detected and restarted after each subsequent reception. Expiration of this timer terminates reception.

Receive Timeout

Defines the receive data timeout time. While its specific application varies somewhat depending on the selected *Receive Termination Mode*, expiration of this timer generally indicates an unsuccessful timeout condition. Refer to section 1.5 for a detailed summary of the Receive Timeout behavior.

Number of Bytes

Specifies the size of the database buffer, starting at the designated *Database Address*. The entire database buffer of size "Number of Bytes" is modified upon every successful reception (either with actual received data or zero-filled data appended to received data). Refer to section 1.5 for further details regarding the application of the Number of Bytes field for each *Receive Termination Mode*.

Database Address

Defines the start of the database buffer, the size of which is specified by the *Number of Bytes* field.

1.5 Packet Transfer and Termination Summary

This section serves as an overview of the reception flow processing action and evaluation criteria used to determine the successful or unsuccessful reception of packetized data for each of the Receive Termination Modes. For TCP connection objects, these processing actions assume that a TCP socket connection has already been established (as no processing takes place outside the context of a TCP connection). For UDP connection objects, these processing actions occur continuously, without any prior requirements.

1.5.1 Termination Sequence

- If a *Transmit Starting Sequence* is in use, it is sent to the server device.
- The *Receive Timeout* timer is started. If the *Receive Timeout* timer expires prior to any data being received, then the processing action terminates with a failed/timeout status.
- If received data is detected, but a packet gap exceeding the *Receive Timeout* time occurs prior to detection of the *Receive Termination Sequence*, then the processing action terminates with a failed/timeout status.
- If received data is detected, but the cumulative number of bytes received exceeds 1460 bytes prior to detection of the *Receive Termination Sequence*, then the processing action terminates with a failed/too much data status.
- If received data is detected, and the *Receive Termination Sequence* is detected without a *Receive Timeout* gap, then a packet is consumed from the receive buffer and the processing action terminates successfully.
    - If the cumulative number of bytes in the packet exceeds the designated *Number of Bytes*, then only the first "*Number of Bytes*" is retained and the remainder is discarded.
    - If the cumulative number of bytes in the packet is less than the designated *Number of Bytes*, then the remainder of the database buffer is zero-filled.
- Note that depending on relative timing and the transmit characteristics of the server device, it may be possible to receive data that includes more than one *Receive Termination Sequence* (multiple packets may exist in the receive buffer). In such a scenario, only the "last" full packet is consumed, and the others are discarded. Any data received after the "last" full packet is assumed to be a partial packet for the next *Receive Termination Sequence*, and is therefore retained for the next processing action.

Example

Receive Termination Sequence = 0D (ASCII “carriage return”)
Strip Receive Termination Sequence enabled

Received data (`N`, `P`, `W`… are any 8-bit values other than 0x0D):

```
N   P   W  0x0D  Y   Z
```

- If *Number of Bytes* = 2, then `NP` is stored in the database buffer
- If *Number of Bytes* = 3, then `NPW` is stored in the database buffer
- If *Number of Bytes* = 5, then `NPW00` is stored in the database buffer
- `YZ` remains in the receive buffer for a subsequent processing action, and will not be packetized until an additional carriage return is received.

1.5.2 Message Timeout

- If a *Transmit Starting Sequence* is in use, it is sent to the server device.
- The *Receive Timeout* timer is started. If the *Receive Timeout* timer expires prior to any data being received, then the processing action terminates with a failed/timeout status.
- If received data is detected, then the *Receive Timeout* timer is stopped and the *Receive Delay* timer is started.
    - o If the cumulative number of bytes received exceeds 1460 bytes prior to the *Receive Delay* timer expiration, then the processing action terminates with a failed/too much data status.
    - o All subsequent data is received until the *Receive Delay* timer expires. A packet is then consumed from the receive buffer and the processing action terminates successfully.
    - o If the cumulative number of bytes in the packet exceeds the designated *Number of Bytes*, then only the first “*Number of Bytes*” is retained and the remainder is discarded.
    - o If the cumulative number of bytes in the packet is less than the designated *Number of Bytes*, then the remainder of the database buffer is zero-filled.

Example

Receive Delay = 1000mS
Scan Rate = 0mS

Received data (`M`, `N`, `P`… are any 8-bit values):

```
  TIME (T) →
| 0                  | 1000mS              | 2000mS
|                    |                     |
   M    N    P    W    X    Y         Z
```

- If *Number of Bytes* = 2, then `MN` is stored in the database buffer at time T=1S, and `XY` is stored in the database buffer at time T=2S
- If *Number of Bytes* = 4, then `MNPW` is stored in the database buffer at time T=1S, and `XYZ0` is stored in the database buffer at time T=2S

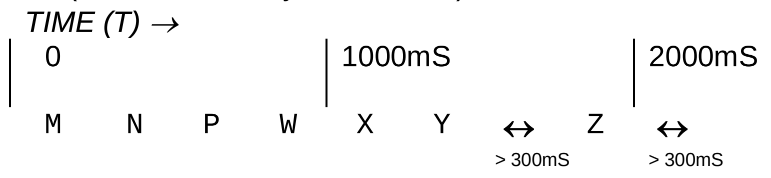
1.5.3 Character/Packet Gap Delay

- If a *Transmit Starting Sequence* is in use, it is sent to the server device.
- The *Receive Timeout* timer is started. If the *Receive Timeout* timer expires prior to any data being received, then the processing action terminates with a failed/timeout status.
- If received data is detected, then the *Receive Timeout* timer is stopped and the *Receive Delay* timer is started.
    - After each data character/packet reception, the *Receive Delay* timer is restarted.
    - If the cumulative number of bytes received exceeds 1460 bytes prior to *Receive Delay* timer expiration, then the processing action terminates with a failed/too much data status.
    - When the *Receive Delay* timer expires (i.e. a packet gap is detected), then a packet is consumed from the receive buffer and the processing action terminates successfully.
    - If the cumulative number of bytes in the packet exceeds the designated *Number of Bytes*, then only the first "*Number of Bytes*" is retained and the remainder is discarded.
    - If the cumulative number of bytes in the packet is less than the designated *Number of Bytes*, then the remainder of the database buffer is zero-filled.

<u>**Example**</u>

Receive Delay = 300mS
Scan Rate = 0mS

Received data (`M`, `N`, `P`… are any 8-bit values):



- If *Number of Bytes* = 4, then `MNPW` is stored in the database buffer after the first gap detection, and `Z000` is stored in the database buffer after the second gap detection.

1.5.4 Fixed Size

- If a *Transmit Starting Sequence* is in use, it is sent to the server device.
- The *Receive Timeout* timer is started. If the *Receive Timeout* timer expires prior to any data being received, then the processing action terminates with a failed/timeout status.
- If received data is detected, but a packet gap exceeding the *Receive Timeout* time occurs prior to receiving the designated *Number of Bytes*, then the processing action terminates with a failed/timeout status.
- If received data is detected, and the designated *Number of Bytes* is received without a *Receive Timeout* gap, then a packet is consumed from the receive buffer and the processing action terminates successfully.

Example

Number of Bytes = 4

Received data (A, B, C… are any 8-bit values):

```
A   B   C   D   E   F   G   H   I   J
```

- Packet ABCD is stored in the database buffer after the first processing action.
- Packet EFGH is stored in the database buffer after the second processing action.
- I and J remain in the receive buffer for a subsequent processing action, and will not be packetized until two additional bytes are received.

1.6 Manual Trigger

1.6.1 Overview

The processing action of every connection object is triggered to execute periodically by some form of stimulus. Once triggered, this processing action entails sending a *Transmit Start Sequence* (if enabled), and managing incoming received data (refer to section 1.5). By default, the driver automatically provides the trigger stimulus based on expiration of the *Scan Rate* timer. However, there may be situations where external (manual) control of this trigger stimulus is desirable in order to achieve a certain degree of control over a connection object's behavior. This can be accomplished by adding an optional Manual Trigger element to the connection object. Once added, a manual trigger can act as the connection object's execution stimulus by manipulating a single bit in the internal database. Trigger bits can be manipulated either by actively injecting data into the database from a remote client via any supported server protocol, or by new data values being actively read into the database via service objects associated with a client protocol.

1.6.2 Manual Trigger Settings

Trigger Address

Specifies the database address that contains the byte-size *Trigger Bit* structure.

Trigger Bit

Specifies which bit in the byte designated by the *Trigger Address* is to be used as the trigger bit. Only one bit may be selected in the *Trigger Bit* structure, and each bit at a given *Trigger Address* may only be associated with one connection object (in other words, there is a unique "one connection object to one trigger bit" association). This mechanism allows up to 8 connection object trigger bits to be simultaneously assigned to any given database address.

1.6.3 Behavior

- When auto trigger is used, cyclic expiration of the *Scan Rate* timer causes the connection object to unconditionally trigger.
- When manual trigger is used, cyclic expiration of the *Scan Rate* timer causes the connection object to only evaluate whether or not it should trigger. This evaluation is performed by inspecting the state of the trigger bit and reacting to it as a "one-shot", meaning that when an external source sets the trigger bit to "1", then the corresponding connection object is

unconditionally triggered. Once the processing action has been completed (either successfully or unsuccessfully), the driver will update the database with the received data, update the diagnostics object data structure (if implemented) to indicate the success or failure of the transaction, and then automatically clear the trigger bit. In this way, a remote device can both trigger an action, as well as be notified of the completion and resulting status by appropriately mapping the trigger bit, connection object data, and diagnostics object structure into a block of data that is accessible via another network (such as a read/write service object controlled by a client driver on another network). If the connection object evaluates its trigger bit to be “0”, then it takes no further action: no data is transmitted to the server device, and any available received data is not processed.

Note that because the internal database is initialized to “0” values after every boot cycle, all defined manual trigger bits will cause their respective connection objects to be disabled until explicitly enabled from an external source.

1.7 Diagnostics Object

Each TCP and UDP connection object can optionally include a diagnostics object for debugging and diagnostics.

Diagnostics Database Address

Enter the database address at which to store the diagnostics information.

INDUSTRIAL CONTROL COMMUNICATIONS, INC.

1600 Aspen Commons, Suite 210
Middleton, WI USA 53562-4720
Tel: [608] 831-1255   Fax: [608] 831-2045

http://www.iccdesigns.com

Printed in U.S.A