Garmin PNA Integration Manual

Proprietary and Confidential

Version 31.1

Revised and Updated: August 7, 2011

Pointer Telocation Ltd. 14 Hamelacha St., Rosh Ha'ayin 48091, Israel • Tel: 972-3-5723111 • Fax: 972-3-5723100 • www.pointer.com

Legal Notices

IMPORTANT

1. All legal terms and safety and operating instructions should be read thoroughly before the product accompanying this document is installed and operated.
2. This document should be retained for future reference.
3. Attachments, accessories or peripheral devices not supplied or recommended in writing by Pointer Telocation Ltd. May be hazardous and/or may cause damage to the product and should not, in any circumstances, be used or combined with the product.

General

The product accompanying this document is not designated for and should not be used in life support appliances, devices, machines or other systems of any sort where any malfunction of the product can reasonably be expected to result in injury or death. Customers of Pointer Telocation Ltd. Using, integrating, and/or selling the product for use in such applications do so at their own risk and agree to fully indemnify Pointer Telocation Ltd. For any resulting loss or damages.

Warranty Exceptions and Disclaimers

Pointer Telocation Ltd. Shall bear no responsibility and shall have no obligation under the foregoing limited warranty for any damages resulting from normal wear and tear, the cost of obtaining substitute products, or any defect that is (i) discovered by purchaser during the warranty period but purchaser does not notify Pointer Telocation Ltd. Until after the end of the warranty period, (ii) caused by any accident, force majeure, misuse, abuse, handling or testing, improper installation or unauthorized repair or modification of the product, (iii) caused by use of any software not supplied by Pointer Telocation Ltd., or by use of the product other than in accordance with its documentation, or (iv) the result of electrostatic discharge, electrical surge, fire, flood or similar causes. Unless otherwise provided in a written agreement between the purchaser and Pointer Telocation Ltd., the purchaser shall be solely responsible for the proper configuration, testing and verification of the product prior to deployment in the field.

POINTER TELOCATION LTD.'S SOLE RESPONSIBILITY AND PURCHASER'S SOLE REMEDY UNDER THIS LIMITED WARRANTY SHALL BE TO REPAIR OR REPLACE THE PRODUCT HARDWARE, SOFTWARE OR SOFTWARE MEDIA (OR IF REPAIR OR REPLACEMENT IS NOT POSSIBLE, OBTAIN A REFUND OF THE PURCHASE PRICE) AS PROVIDED ABOVE. POINTER TELOCATION LTD. EXPRESSLY DISCLAIMS ALL OTHER WARRANTIES OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, SATISFACTORY PERFORMANCE AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL POINTER TELOCATION LTD. BE LIABLE FOR ANY INDIRECT, SPECIAL, EXEMPLARY, INCIDENTAL OR CONSEQUENTIAL DAMAGES (INCLUDING WITHOUT LIMITATION LOSS OR INTERRUPTION OF USE, DATA, REVENUES OR PROFITS) RESULTING FROM A BREACH OF THIS WARRANTY OR BASED ON ANY OTHER LEGAL THEORY, EVEN IF POINTER TELOCATION LTD. HAS BEEN ADVISED OF THE POSSIBILITY OR LIKELIHOOD OF SUCH DAMAGES.

Intellectual Property

Table of Contents

1 Introduction

This document describes the integration of the Garmin interface into the infrastructure of the CCC (Command and Control Center). The integration provides the ability to remotely control a Garmin PNA (Personal Navigation Assistant) via the FW unit.

1.1 Main Features

Using the provided interface the customer is able to perform the following:

- Send different types of a text messages (just text, Yes/No, OK) to a Garmin terminal and to register the response.
- Send stop points (destinations) to a Garmin terminal so that it can navigate to these stop points.
- Obtain ETA to a specific stop.
- Delete stop points from the Garmin terminal's memory.
- Erase all messages from the Garmin terminal's memory.

1.2 References

No.	Document Name	Version	Date	Remark
1	FW Wireless Protocol	v27 p&u	24/06/08	Or newer
2	001-00096-00	Rev A	1/01/07	Garmin's API document (or newer)
3	001-00096-00	Rev D	22/09/08	
4	FW28 Release Notes	1	01/01/09	Or newer

1.3 Revisions

Version	Date	Description
28.0	14/01/09	Original version.
28.1	15/1/09	The CU is sending "Enable Fleet management mode" commands to the Garmin Terminal every 15 seconds. In packet type 5: the spare byte is a notification that the payload should be forwarded to Serial interface in Garmin format. In packet type 8: After authentication code "Forwarded message code" added. "Packet from Garmin" removed.

Version	Date	Description
28.2	17/1/09	Support of Legacy Text protocol removed. Minor typo fixes. Links to Garmin's API and Supported devices web pages added.
28.3	20/1/09	OTA Message type 5: Garmin data length in limited to 199 bytes (186 bytes of text to Garmin). OTA message type 8: Settings byte will always be sent as "1". Garmin data length field changed to 2 bytes.
28.4	5/2/09	Added application note concerning the terminals, which report A604 as part of their protocol support data.
28.5	19/4/09	Zumo 400 deleted from the list of supported devices.
28.6	26/4/09	List of supported terminals updated.
28.7	3/9/09	DLE stuffing covered.
28.8	8/10/09	Description of OTA ACK corrected: Integration overview and Data Flow Chart.
28.9	3/12/09	Added new supported Garmin text protocols and canned messages and list protocols as per 001-00096-00 Rev D.
30.0	04/04/09	Updating section 1.5. Updated links to Garmin site.
31.1	7/8/11	Fixed new and replaced interface cables. Updated integration overview. Added programming parameters overview.

1.4 Abbreviations

Abbreviation	Definition
ACK	Acknowledge
CCC	Command and Control Center
CU	Cellocator unit
ETA	Estimated time of arrival
OTA	Over the air
PNA	Personal Navigation Assistant
PND	Personal Navigation Device

1.5 Supported Garmin Terminals and Required Data Cables

The following link to Garmin web site described all Garmin products which support the Garmin protocol and the required accessory for each one of them.

Link to Garmin's supported products

The following Fleet Management Interface Cables are used for connecting Garmin devices to the Cellocator device / harness.

1. 5V+Serial Pigtail to Mini B

Part Number: 010-11627-00 (replaces the previous FMI 10 cable (010-11232-00))

2. Fleet Management Interface Cable

Part Number: 010-10865-00

3. Data Cable (Mini-B to Serial Pigtail)

Part Number: 010-10813-00

4. FMI 40 Cable (Data & Traffic)

Part Number: 010-11628-00

This cable replaces previous FMI 40 cable (010-11259-00).

2 General Description

2.1 General Overview

The Garmin PNA serves both as a navigator and a mobile data terminal in the vehicle, while the Cellocator unit provides seamless connectivity to the monitoring center via a wireless mobile data network.

Garmin's fleet management and dispatch messaging interface enables direct communication via text messaging, as well as instant re-routing with "new destination" message prompts.

2.2 Integration Overview

1. The “Enable Garmin compatibility mode” bit in EEPROM should be set in order to enable Garmin support.
2. If Garmin is enabled in the unit's configuration, the unit in Standby Mode sends the Garmin “Enable fleet management protocol” every 15 seconds until the Garmin responds for the first time.
3. Upon reception of the response from the Garmin, the unit:
    a) Generates and sends a "Garmin Connected" OTA event or distress, as per programming.
    b) Updates the OTA "Garmin Connected" flag byte in every outbound message.
    c) Initiates a Product ID Request to the Garmin.
    d) Forwards from the Garmin to the CCC all the response packets: Product ID product_id_data_typ, Protocol Support Data, and any other packets generated by Garmin as a response to the Product ID Request.
4. Once the unit detects three missing responses from the Garmin, it:
    a) Generates a "Garmin Disconnected" OTA event or distress, as per programming.
    b) Updates the OTA "Garmin Connected" flag in outbound OTA messages.
5. From the moment of Garmin disconnection, the CU continues sending the Garmin “Enable fleet management" message every 15 seconds until the Garmin responds or enters hibernation.
6. Every message sent between the Garmin and the CU is acknowledged by the receiving side. The CCC is not exposed to this internal communication and should not take any active part in it.
7. Command forwarding to the Garmin is based on the Forward Data OTA command (type 5 of the outbound channel of Cellocator Wireless protocol).
8. Messages received from the Garmin are forwarded to the wireless channel as a payload of a standard packet type 8 (inbound channel of Cellocator Wireless protocol).
9. The packet includes a notification bit stating that the message was delivered from the attached Garmin device.

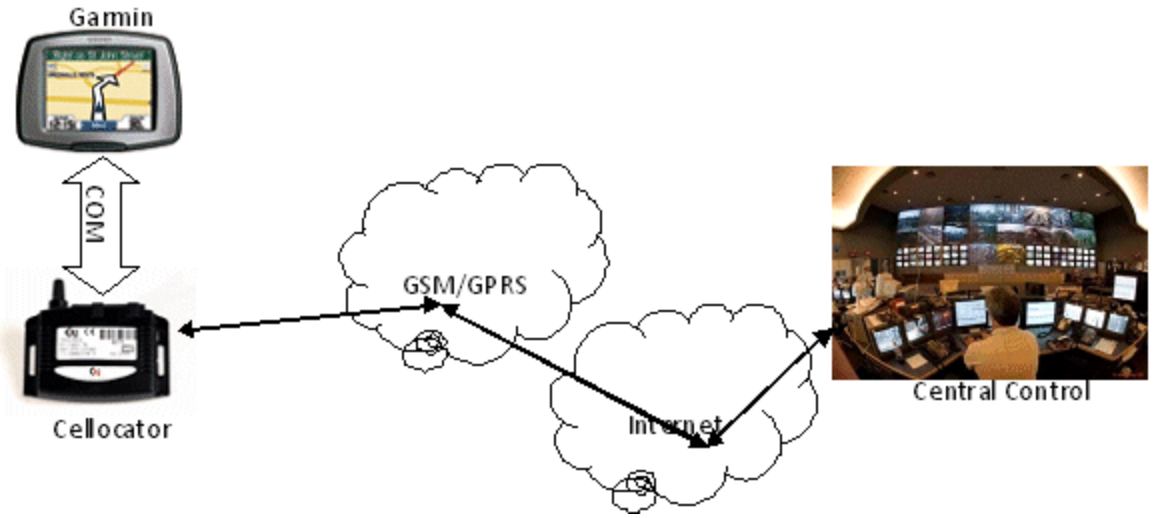
10. Long messages (longer than 74 characters) from a Garmin are split into a number of packets.
11. The unit delivers the Garmin data via an SMS message when a GPRS connection is not available (optional).
12. Packets coming from the Garmin require a Cellocator protocol ACK (Packet type 4, except in cases when they are received via SMS), and some of the messages delivered from a Garmin require an application ACK, which should be generated by the CCC application. Refer to the supported Garmin application protocols listed later in this document.
13. It is possible to configure the CU not to send an ACK to the Garmin terminal in cases of GSM connectivity loss. In such a scenario, the Garmin will repeat the same message again and again until GSM coverage is restored, and those repetitions will not be stored in the CU's buffer.

    Otherwise the Gamin terminal will resend messages upon lack of an ACK from the far server and will fill the CU buffer with useless repetitions of the same message.
14. It is possible to configure the CU to support older terminals not supporting the "PING" command:
    a) If "backward compatible mode" is enabled in programming, the CU pings the Garmin Terminal using "Enable fleet management protocol".
    b) If "backward compatible mode" is disabled in programming, the CU pings the Terminal using the Ping protocol.

3 Communication

This section describes the communication layer which enables the delivery of commands to the Garmin interface and to receive and de-capsulate a packet received from a Garmin.

3.1 System Chart

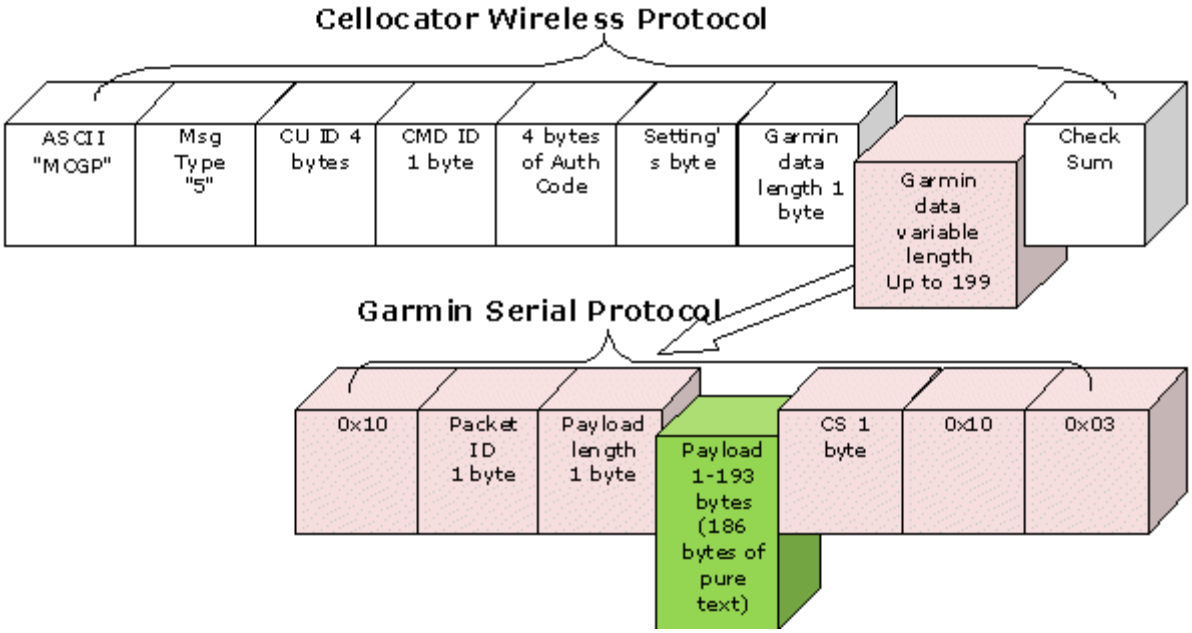


3.2 Data Flow Chart

CCC	Cellocator inbound packet (type 5) Forwarding a command to Garmin	Cellocator unit	Garmin, connected to a COM port of Cellocator unit
	Cellocator outbound packet (type 0) Can be programmatically disabled (address in EERPOM 0x544, bit 0)		
	Cellocator outbound packet (type 8) Forwarding a reply from Garmin		
	Cellocator inbound packet (Type 4) – Generic Acknowledge (ACK) Required for every type 8 message received via GPRS. Type 8 message received via SMS does not require ACK.		
	Cellocator inbound packet (type 5) Forwarding an application ACK (receipt) to Garmin (required in some of the cases, see below).		
	Cellocator outbound packet (type 0) Can be programmatically disabled (address in EERPOM 0x544, bit 0)		

3.3 Encapsulation into Cellocator Wireless Protocol

3.3.1 *Packet to Garmin (type 5)*



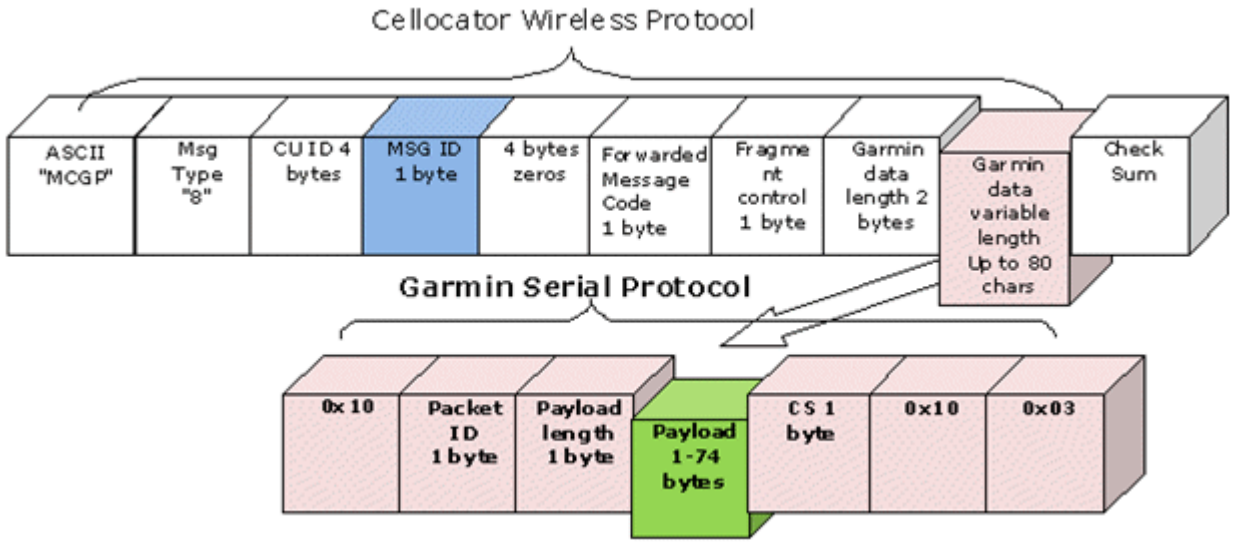
3.3.2 *Setting's Byte*

This byte will be used for different system indications.

Reserved, should be sent as zero							Packet to Garmin (compliant to Garmin serial interface)
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

Packet to Garmin is set to"1" if the packet is to be forwarded to a Garmin terminal.

3.3.3 *Packet from Garmin (type 8)*



- **Forwarded Message Code:** A counter of forwarded messages is updated every time message is forwarded from the terminal (currently always sent to "1").
- **Fragment Control Byte:** If the standard 74-byte length does not meet the requirements, the message from a Garmin will be forwarded in two or more fragments.

Unit Originated (* Always 1)	Last Fragment	Fragment Index					
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

- **Last Fragment Bit** – set to "1" if the fragment is the last one.
- **Unit Originated Bit** – set to "1" if the message origin is a unit and "0" if the message origin is an application. Currently for Cellocator Server compatibility reasons it is always sent as "1".
- **Fragment Index** – fragment number.

3.3.4 *DLE Stuffing*

According to Garmin's API document (001-00096-00), if any byte in the Garmin Size, Packet Data, or Checksum fields is equal to DLE (0x10), then a second DLE shall be inserted immediately following the byte.

This extra DLE is not included in the size or checksum of the Garmin payload calculation. This procedure allows the DLE character to be used to delimit the boundaries of a Garmin packet.

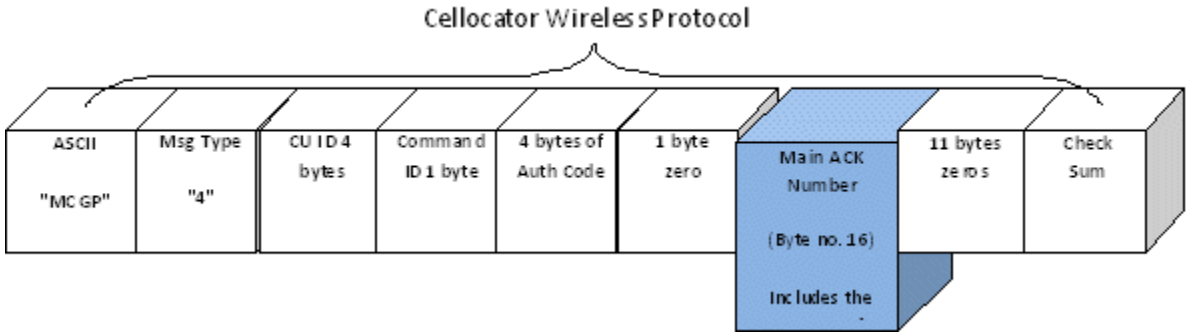
Cellocator unit will detect DLE Stuffing in the messages received from a Garmin terminal, will remove the extra DLE from the payload and forward the "clear" payload to the server.

The Cellocator unit will NOT add the DLE Stuffing to the messages received from the server; the DLE Stuffing shall be managed on server upon generation of the message to Garmin terminal.

The Cellocator unit is not checking the syntax of Garmin's payload and will forward the message as it is received from the server. The terminal will reply by NACK to an invalid message, although this NACK will not be forwarded to the server by the Cellocator unit.

3.3.5 *Generic ACK (type 4) – Cellocator Protocol Level*

Sent to Cellocator unit as a reception confirmation for message type 8, it is always sent when a message of type 8 is received over GPRS. A type 8 message received via SMS does not require an ACK.



3.3.6 *Garmin Status in Message Type 0 (Outbound Channel of Cellocator Wireless protocol)*

First byte of Communication Control (10th):

Normalized CAN originated Odometer data	Normalized CAN originated Speed data	Dallas field association bits		Message source	Garmin Connected	Garmin Enabled	Message Initiative
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

- The "Garmin Enabled" field monitors the status of a corresponding bit in the configuration memory (1 when enabled).
- The "Garmin Connected" bit monitors the status of a communication between the Garmin and the CU. This bit is set with the first correct ACK or NACK received from the Garmin and reset upon three missing responses from the Garmin (timeout expiration). The CU will automatically reset this bit upon entering any type of Hibernation.

4 Programmable Parameters

4.1 Enable Garmin Support

Address 1348 bit 2

Description: Enables detecting Garmin upon connection and disconnection and updating accordantly, forwarding data from and to the terminal

Default value: 0 - Disabled

4.2 Discard Garmin packets during NoIP/NoGSM

Address: 285, bit 5

Description: In order to prevent duplication of Garmin messages in unit's COM buffer (and as result duplication in server's DB) during lack of IP connection, it is possible to discard (without storing in the buffer) any Garmin packet received:

- When no IP connection to the operational server is available
- When no GSM registration is available (including modem off periods) - (If this bit is "0" AND "Data forwarding by SMS" on addresses 202/204 bit 5 is enabled (1))
- During maintenance session

Default value: "0" – discard

4.3 Enable Garmin Connected/Disconnected Messages

Address 285 bit 0 for Event

285 bit 1 for Distress

Description: Once this bit is enabled the unit will generate an appropriate message to CCC upon Garmin terminal connection and disconnection.

Transmission Reason	Specific Transmission Reason
205 Garmin connection status changed	0 – Garmin Disconnected 1 – Garmin Connected

Default value: 0 - Disabled

4.4 Disable Garmin's Ping (0x260) upon Detection of the Terminal

Address 285 bit 2

Description: If this bit is enabled, the CU will immediately disable Ping from the Garmin to the CCC upon detection of the terminal's connection (by sending to the Garmin Message Throttling Command Packet 0x0250; the Garmin's response to this command 0x0251 will NOT be forwarded to the server).

N	Direction	Fleet Management Packet ID	Fleet Management Packet Data Type
0	Client to Server	0x0260 – Ping Packet ID	None
1	Server to Client	0x0261 – Ping Response Packet ID	None

Note that this protocol is disabled by default in terminals reporting A605 as part of their protocol support data.

This protocol is not supported at all by terminals not reporting A604 as part of their protocol support data.

Default value: 0 – Disable Ping

4.5 Backward Compatible Mode

Address 285 bit 3

Description: If enabled, the unit will ping the Garmin using the "0x0000 Enable fleet management protocol" instead of 0x0260 – Ping Packet ID. This is for older Garmin terminals, not reporting A604 as part of their protocol support data.

Default value: 0 - Disabled

5 Supported Garmin Packets

[Download Fleet Management Interface Developer Kit]

5.1 Packet Sequences

Each of the application protocols is defined in terms of a packet sequence, which defines the order and types of packets exchanged between two devices, including the direction of the packet, the packet ID, and the packet data type. An example of a packet sequence is shown in the following table.

N	Direction	Packet ID	Packet Data Type
0	CCC to Garmin	First_Packet_ID	First_Data_Type
1	CCC to Garmin	Second_Packet_ID	Second_Data_Type
2	CCC to Garmin	Third_Packet_ID	Third_Data_Type
3	Garmin to CCC	Fourth_Packet_ID	Fourth_Data_Type
4	Garmin to CCC	Fifth_Packet_ID	Fifth_Data_Type

In this example, there are five packets that are exchanged: three from the CCC to the Garmin and two from the Garmin to the CCC.

The first column shows the packet number (used only for reference; this number is not encoded into the packet).

5.2 Undocumented Application Packets

The Garmin may transmit application packets containing packet IDs that are not documented in this specification.

These packets are used for internal testing purposes by Garmin engineers. Their contents are subject to change at any time and should not be used by third-party applications for any purpose. They should be handled according to the physical/link protocols described in this specification and then discarded.

5.3 Garmin Application Protocols

5.3.1 *Fleet Management Protocols*

- **Protocol Identifier:** All packets related to the fleet management protocols use the packed ID 161. The first 16-bits of data in the application payload identify the fleet management protocol's specified packet. The remaining data in the application payload is the fleet management payload. The fleet management data types discussed in this document do not include the fleet management packet ID in their structures. The fleet management packet ID is implied. The following table shows the format of a fleet management packet.

Notes	Byte Description	Byte Number
16 (decimal)	Data Link Escape	0

Notes	Byte Description	Byte Number
161 (decimal)	Packet ID	1
Size of fleet management Payload + 2	Size of Packet Data	2
See Application Note	Fleet management Packet ID	3-4
0 to 253 bytes	Fleet management Payload	5 to n-4
2's complement of the sum of all bytes from byte 1 to byte n-4	Checksum	n-3
16 (decimal)	Data Link Escape	n-2
3 (decimal)	End of Text	n-1

5.3.2 *Product ID and Support Protocol*

The Product ID and supported protocol are used to query the Garmin to find out its Product ID, software version number, and supported protocols and data types. The packet sequence for the Product ID and support protocol is shown in the following table.

Fleet Management Packet ID	Fleet Management Packet Data Type	Direction	N
0x0001 – Product ID Request	None	CCC to Garmin	0
0x0002 – Product ID	Product_ID_Data_Type	Garmin to CCC	1
0x0003 – Protocol Support Data	Protocol_Array_Data_Type	Garmin to CCC	2

The type definition for the Product_ID_Data_Type is shown below.

```
typedef struct
{
uint16    product_id;
sint16    software_version;
} Product_ID_Data_Type;
```

product_id is a unique number given to each type of Garmin device. It should not be confused with an ESN, which is unique to each device regardless of type. software_version is the software version number multiplied by 100 (e.g. version 3.11 will be indicated by 311 decimal).

The type definition for the Protocol_Support_Data_Type is shown below. The Protocol_Array_Data_Type is an array of the Protocol_Support_Data_Type. The number of Protocol_Support_Data_Type contained in the array is determined by observing the size of the received packet data.

```
typedef struct
{
char      tag;
sint16    data;
} Protocol_Support_Data_Type;
```

The tag member can have different values. The A tag describes an "application" protocol. For example, if a Garmin reports a tag with an "A" and data of 602 (A602), then it supports some of the fleet management protocol defined in this document. Each section in this document that describes a protocol will state its support A tag.

The D (data type) tags that are listed immediately after the A (application protocol) tag describe the specific data types used by that application protocol. This allows for future growth of both the data types and the protocol. For example, if a Garmin reports a tag with a "D" and data of 602, then it supports some of the fleet management data types defined in this document. Each section in this document that describes a data type will state its support D tag.

The CCC is expected to communicate with a Garmin using the Garmin's stated application protocol and data types. In this way, it is possible to have different generations of products on the field and still have a workable system.

5.4 Text Message Protocols

5.4.1 *CCC to Garmin Text Message Protocols*

There are numerous protocols available to the CCC for sending text messages to a Garmin. The CCC selects a text message protocol to send a text message to a Garmin depending on the type of functionality it is trying to achieve on the Garmin. The different types of CCC-to-Garmin text message protocols are described in detail below.

5.4.1.1 A604 CCC to Garmin Open Text Message Protocol

This text message protocol is used to send a simple text message from the server to the client. When the client receives this message, it either displays the message immediately, or presents a notification that a message was received, depending on the options specified in the message. The receipt indicates whether the message was accepted by the device; it does not imply that the message has been displayed.

This protocol is only supported on clients that report A604 as part of their protocol support data. The packet sequence for the server to client open text message is shown below:

Fleet Management Packet Data Type	Fleet Management Packet ID	Direction	N
A604_CCC_to_Garmin_open_text_ msg_data_type	0x002a – A604 CCC to Garmin Open Text Message Packet ID	CCC to Garmin	0
server_to_client_text_msg_receipt_data_type	0x002b – CCC to Garmin Open Text Message Receipt Packet ID	Garmin to CCC	1

The type definition for the A604_server_to_client_open_text_msg_data_type is shown below. This data type is only supported on clients that report D604 as part of their protocol support data.

```
typedef struct /* D604 */
{
time_type origination_time;
uint8 id_size;
uint8 message_type;
uint16 reserved; /* set to 0 */
uint8 id[/* 16 bytes */];
uchar_t8 text_message[/* variable length, null-terminated string, 200 bytes max
*/];
} A604_server_to_client_open_text_msg_data_type;
```

The origination_time is the time that the text message was sent from the server. The id_size determines the number of characters used in the id member. An id size of zero indicates that there is no message id. The id member is an array of 8-bit integers that could represent any type of data. A message ID is required to use the Message Status Protocol (see below).The message_type indicates how the message should be handled on the client device.

The allowed values for message_type are:

Value (Decimal)	Acknowledgment Type
0	Add message to inbox, and display a floating button indicating that a message was received.
1	Display the message on the device immediately.

The type definition for the open_text_msg_receipt_data_type is shown below. This data type is only supported on clients that report D604 as part of their protocol support data.

```
typedef struct /* D604 */
{
time_type origination_time;
uint8 id_size;
boolean result_code;
uint16 reserved; /* set to 0 */
uint8 id[/* 16 bytes */];
} server_to_client_text_msg_receipt_data_type;
```

The origination_time, id_size, message_type, and id will be the same as the corresponding A604 Server to Client Open Text Message packet. The result_code indicates whether the message was received and stored on the client device; it will be true if the message was accepted, or false if an error occurred (for example, there is already a message with the same ID).

5.4.1.2 Server to Client Canned Response Text Message Protocol

This text message protocol is used to send a text message from the server to the client who requires a response to be selected from a list. When the message is displayed, the client will also display a Reply button. When the Reply button is pressed, the client will display a list of the allowed responses. Once the user selects one of the responses, the client will send an acknowledgement message to the server indicating which response was selected.

Prior to using this protocol, the server must send the text for allowed responses to the client using the Canned Response List Protocols described in section below. This protocol is only supported on clients that report A604 as part of their protocol support data. The packet sequence for the Server to Client Canned Response Text Message protocol is shown below:

Fleet Management Packet Data Type	Fleet Management Packet ID	Direction	N
canned_response_list_data_ type	0x0028 – Canned Response List Packet ID	CCC to Garmin	0
canned_response_list_receipt_ data_type	0x0029 – Canned Response List Receipt Packet ID	Garmin to CCC	1
A604_CCC_to_Garmin_open_ text_ msg_data_type	0x002a - A604 Server to Client Open Text Message Packet ID	CCC to Garmin	2
text_msg_ack_data_type	0x0020 – Text Message Acknowledgment Packet ID	Garmin to CCC	3
text_msg_id_data_type	0x002c – Text Message Ack Receipt Packet ID	CCC to Garmin	4

The type definition for the canned_response_list_data_type is shown below. This data type is only supported on clients that report D604 as part of their protocol support data.

```
typedef struct /* D604 */
{
uint8 id_size;
uint8 response_count;
uint16 reserved; /* set to 0 */
uint8 id[/* 16 bytes */];
uint32 response_id[];
} canned_response_list_data_type;
```

The id_size and id are used to correlate the canned response list to the A604 Server to Client Open Text Message that follows; the id must be unique for each message sent to the device, and cannot have a length of zero. The response_id array contains the IDs for

up to 50 canned responses that the user may select from when responding to the message. The response_count indicates the number of items in the response_id array.

The type definition for the canned_response_list_receipt_data_type is shown below. This data type is only supported on clients that report D604 as part of their protocol support data.

```
typedef struct /* D604 */
{
uint8 id_size;
uint8 result_code;
uint16 reserved; /* set to 0 */
uint8 id[/* 16 bytes */];
} canned_response_list_receipt_data_type;
```

The id_size and id will be identical to those received using the Canned Response List packet. A result_code of zero indicates success, a nonzero result_code means that an error occurred, according to the table below. Note that the protocol should not continue if the result_code is nonzero.

Result Code (Decimal)	Meaning	Suggested Response
0	Success (This is same behavior used for the A602 and A603 Server to Client text messages)	Send the text message packet.
1	Invalid response_count	Send a Canned Response List packet containing between1 and 50 response_id entries.
2	Invalid response_id	Use the Set Canned Response protocol (section below) to ensure all of the canned responses are on the client, then resend the Canned Response List packet.
3	Invalid or duplicate message ID	Send a Canned Response List packet using a message ID that is not on the client.
4	Canned Response List database full	Wait for the Text Message Acknowledgement packet from a previous Server to Client Canned Response Text Message, then restart the protocol.

The packet ID and type definition for the A604 Server to Client Open Text Message are described in section above. The type definition for the text_msg_ack_data_type is shown below. This data type is only supported on clients that report D602 or D604 as part of their protocol support data.

```
typedef struct /* D602/D604 */
{
time_type origination_time;
uint8 id_size;
uint8 reserved[3]; /* set to 0 */
uint8 id[/* 16 bytes */];
uint32 msg_ack_type
} text_msg_ack_data_type;
```

The origination_time is the time that the text message was acknowledged on the client. The id_size and id will match the id_size and id of the applicable text message. The msg_ack_type will identify the response_id corresponding to the response selected by the user.

The type definition for the text_msg_id_data_type is shown below. This data type is only supported on clients that report D604 as part of their protocol support data.

```
typedef struct /* D604 */
{
uint8 id_size;
uint8 reserved[3]; /* set to 0 */
uint8 id[/* 16 bytes */];
} text_msg_id_data_type;
```

The id_size and id will match the id_size and id of the applicable text message.

5.4.1.3 Message Status Protocol

The Message Status Protocol is used to notify the server of the status of a text message previously sent from the server to the client. The client will send a message status packet whenever the status changes. If the protocol is throttled (see section 5.1.16), the client will only send the message status of a text message when it is requested by the server.

This protocol is only supported on clients that report A604 as part of their protocol support data. The packet sequence for the client to server open text message is shown below:

Fleet Management Packet Data Type	Fleet Management Packet ID	Direction	N
message_status_request_data_type	0x0040 – Message Status Request Packet ID0x0040 – Message Status Request Packet ID	CCC to Garmin	0
message_status_data_type	0x0041 – Message Status Packet ID	Garmin to CCC	1
message_status_data_type	0x0041 – Message Status Packet ID	Garmin to CCC	0

The type definition for the message_status_request_data_type is shown below. This data type is only supported on clients that report D604 as part of their protocol support data.

```
typedef struct /* D604 */
{
uint8 id_size;
uint8 reserved[3]; /* set to 0 */
uint8 id[/* 16 bytes */];
} message_status_request_data_type;
```

The id_size and id correspond to those of the message being queried; all must match for the message to be found.

The type definition for the message_status_data_type is shown below. This data type is only supported on clients that report D604 as part of their protocol support data.

```
typedef struct /* D604 */
{
uint8 id_size;
uint8 status_code;
uint16 reserved; /* set to 0 */
uint8 id[/* 16 bytes */];
} message_status_data_type;
```

The id_size and id will be identical to those of the corresponding Message Status Request. The status_code indicates the status of the message on the device. The following table shows the possible values for status_code, along with the meaning of each value:

Status Code (Decimal)	Meaning
0	Message is unread
1	Message is read
2	Message is not found (e.g., deleted)

5.4.1.4 Canned Response List Protocols

These protocols are used to maintain the list of canned responses used in the Server to Client Canned Response Text Message Protocol (section above).

Up to 200 canned responses may be stored on the client, and up to 50 of these responses may be specified as allowed for each text message. Canned responses are stored permanently across power cycles.

Set Canned Response Protocol

This protocol is used to set (add or update) a response in the canned response list.

This protocol is only supported on clients that report A604 as part of their protocol support data. The packet sequence for the Set Canned Response Protocol is shown below:

Fleet Management Packet Data Type	Fleet Management Packet ID	Direction	N
canned_response_data_type	0x0030 – Set Canned Response Text Packet ID	CCC to Garmin	0
canned_response_receipt_data_type	0x0032 – Set Canned Response Receipt Packet ID	Garmin to CCC	1

The type definition for the canned_response_data_type is described below. This data type is only supported on clients that report D604 as part of their protocol support data.

```
typedef struct /* D604 */
{
uint32 response_id;
uchar_t8 response_text[]; /* variable length, null terminated, 50 bytes max */
} canned_response_data_type;
```

The response_id is a number identifying this response. If a response with the specified response_id already exists on the device, the response text will be replaced with the response_text in this packet.

The type definition for the canned_response_receipt_data_type is shown below. This data type is only supported on clients that report D604 as part of their protocol support data.

```
typedef struct /* D604 */
{
uint32 response_id;
boolean result_code;
uint8 reserved[3]; /* Set to 0 */
} canned_response_receipt_data_type;
```

The response_id will be the same as that of the corresponding canned_response_data_type. The result_code indicates whether the add/update operation was successful.

Delete Canned Response Protocol

This protocol is used to remove a canned response text from the client device. When a canned response is deleted, it is also removed from the list of allowed responses for any canned response text messages that the client has not replied to. If all allowed responses are removed from a message, the message is treated as a Server to Client Open Text Message.

This protocol is only supported on clients that report A604 as part of their protocol support data.

The packet sequence for the Delete Canned Response Protocol is shown below:

Fleet Management Packet Data Type	Fleet Management Packet ID	Direction	N
canned_response_data_type	0x0031 – Delete Canned Response Packet ID	CCC to Garmin	0
canned_response_receipt_data_type	0x0033 – Delete Canned Response Receipt Packet ID	Garmin to CCC	1

The type definition for the canned_response_delete_data_type is shown below. This data type is only supported on clients that report D604 as part of their protocol support data.

```
typedef struct /* D604 */
{
uint32 response_id;
} canned_response_delete_data_type;
```

The response_id identifies the response to be deleted.

The type definition for the canned_response_receipt_data_type is described in section above and repeated below.

This data type is only supported on clients that report D604 as part of their protocol support data.

```
typedef struct /* D604 */
{
uint32 response_id;
boolean result_code;
uint8 reserved[3]; /* Set to 0 */
} canned_response_receipt_data_type;
```

The response_id will be the same as that of the corresponding canned_response_delete_data_type. The result_code indicates whether the delete operation was successful. This will be true if the response was removed or the response did not exist prior to the operation; it will be false if the canned response cannot be removed.

Refresh Canned Response Text Protocol

This protocol is initiated by the client to request updated response text for a particular message, or for all messages. This protocol is only supported on clients that report A604 as part of their protocol support data, and is throttled by default on clients that report A605 as part of their protocol support data. See the Message Throttling Protocols (see above) to enable the Refresh Canned Response Text Protocol on these clients.

The packet sequence for the Refresh Canned Response Protocol is shown below:

Fleet Management Packet Data Type	Fleet Management Packet ID	Direction	N
request_canned_response_list_refresh_data_type	0x0034 – Request Response Text Refresh Packet ID	CCC to Garmin	0
	(Set Canned Response protocols)		1..n

The request_canned_response_list_refresh_data_type is shown below. This data type is only supported on clients that report D604 as part of their protocol support data.

```
typedef struct /* D604 */
{
uint32 response_count;
uint32 response_id[];
} request_canned_response_list_refresh_data_type;
```

The response_count indicates the number of responses that are in the response_id array. This will always be between 0 and 50. The response_id array contains the response IDs that should be sent by the server using the Set Canned Response protocol.

If response_count is zero, the server should initiate a Set Canned Response protocol for all valid response IDs. If response_count is greater than zero, the server should initiate a Set Canned Response protocol for each response ID in the array, so long as the response ID is still valid.

5.4.1.5 Canned Message (Quick Message) List Protocols

The canned message list maintenance protocols are used to maintain the list of canned (predefined) text messages that a client device may send to the server using the Quick Message feature. When sending a canned message, the user of the client device has the option to modify the text before sending it. The message is sent from the client to the server using the Client to Server Open Text Message Protocol described in section above.

Up to 120 canned messages may be stored on the client. Canned messages are stored permanently across power cycles.

Set Canned Message Protocol

The Set Canned Message Protocol is used to add or update the text of a canned message on the client.

This protocol is only supported on clients that report A604 as part of their protocol support data.

The packet sequence for the Set Canned Message Protocol is shown in the following table.

Fleet Management Packet Data Type	Fleet Management Packet ID	Direction	N
canned_message_data_type	0x0050 – Set Canned Message Packet ID	CCC to Garmin	0
canned_message_receipt_data_type	0x0051 – Set Canned Message Receipt Packet ID	Garmin to CCC	1

The type definition for the canned_message_data_type is shown below. This data type is only supported on clients that report D604 as part of their protocol support data.

```
typedef struct /* D604 */
{
uint32 message_id; /* message identifier */
uchar_t8 message[]; /* message text, null terminated (50 bytes max) */
} canned_message_data_type;
```

The message_id is a number identifying this message. The message_id is not displayed on the client, but it is used to control the order in which the messages are displayed: messages are sorted in ascending order by id. If a message with the specified message_id already exists on the device, it will be replaced with the message text in this packet; otherwise, the message will be added.

The type definition for the canned_message_receipt_data_type is shown below. This data type is only supported on clients that report D604 as part of their protocol support data.

```
typedef struct /* D604 */
{
uint32 message_id; /* message identifier */
boolean result_code; /* result (true if successful, false otherwise) */
uint8 reserved[3]; /* Set to 0 */
} canned_message_receipt_data_type;
```

The message_id is the same number in the corresponding canned_message_data_type. The result_code indicates whether the add/update operation was successful.

Delete Canned Message Protocol

The Delete Canned Message Protocol is used to delete a canned message from the client. This protocol is only supported on clients that report A604 as part of their protocol support data. The packet sequence for the Delete Canned Message Protocol is shown below:

Fleet Management Packet Data Type	Fleet Management Packet ID	Direction	N
canned_message_delete_data_type	0x0052 – Delete Canned Message Packet ID	CCC to Garmin	0
canned_message_receipt_data_type	0x0053 – Delete Canned Message Receipt Packet ID	Garmin to CCC	1

The type definition for the canned_message_delete_data_type is shown below. This data type is only supported on clients that report D604 as part of their protocol support data.

```
typedef struct /* D604 */
{
uint32 message_id; /* message identifier */
} canned_message_delete_data_type;
```

The message_id is a number identifying the message to be deleted.

The type definition for the canned_message_receipt_data_type is shown below. This data type is only supported on clients that report D604 as part of their protocol support data.

```
typedef struct /* D604 */
{
uint32 message_id; /* message identifier */
boolean result_code; /* result (true if successful, false otherwise) */
uint8 reserved[3]; /* Set to 0 */
} canned_message_receipt_data_type;
The message_id is the same number in the corresponding
```

The message_id is the same number in the corresponding canned_message_delete_data_type. The result_code indicates whether the delete operation was successful; this will be true if the specified message was successfully deleted or was not on the device.

Refresh Canned Message List Protocol

The Refresh Canned Message List Protocol is initiated by the client when it requires an updated list of canned messages. In response to this packet, the server shall initiate a Set Canned Message protocol for each message that should be on the client.

This protocol is only supported on clients that report A604 as part of their protocol support data, and is throttled by default on clients that report A605 as part of their protocol support data. See the Message Throttling Protocols (section above) to enable the Refresh Canned Message List Protocol on these clients.

The packet sequence for the Refresh Canned Message Protocol is shown below:

Fleet Management Packet Data Type	Fleet Management Packet ID	Direction	N
None	0x0054 – Refresh Canned Message List Packet ID	Garmin to CCC	0
	(Set Canned Response protocols)		1..n

5.4.1.6 CCC to Garmin Open Text Message Protocol (Deprecated)

This text message protocol is used to send a simple text message from the CCC to the Garmin. When the Garmin receives this message, it displays the text message to the user and no additional action is required from the Garmin after receiving the text message. This protocol is only supported on Garmins that report A602 as part of their protocol support data. The packet sequence for the CCC to Garmin open text message is shown below:

Fleet Management Packet Data Type	Fleet Management Packet ID	Direction	N
CCC_to_Garmin_open_text_msg_data_type	0x0021 – CCC to Garmin Open Text Message Packet ID	CCC to Garmin	0

The type definition for the CCC_to_Garmin_open_text_msg_data_type is shown below. This data type is only supported on Garmin terminals that report D602 as part of their protocol support data.

```
typedef struct /* D602 */
{
time_type origination_time;
char      text_message[/* variable length, null-terminated string, 200 characters max */];
} CCC_to_Garmin_open_text_msg_data_type;
```

The origination_time is the time that the text message was sent from the CCC.

5.4.1.7 CCC to Garmin Simple Okay Acknowledgement Text Message Protocol (Deprecated)

This text message protocol is used to send a simple okay acknowledgement text message from the CCC to the Garmin terminal. When the Garmin receives this message, it displays the text message to the user. It also displays an "Okay" button that the user is required to press after reading the text message. Once the "Okay" button is pressed, the Garmin will send an "Okay" acknowledgement message to the CCC. This protocol is only supported on Garmins that report A602 as part of their protocol support data. The packet sequence for the CCC-to-Garmin simple okay acknowledgement text message is shown below:

Fleet Management Packet Data Type	Fleet Management Packet ID	Direction	N
CCC_to_Garmin_ok_ack_text_msg_data_type	0x0022 – CCC to Garmin Simple Okay Acknowledgment Text Message Packet ID	CCC to Garmin	0
text_msg_ack_data_type	0x0020 – Text message Acknowledgment Packet ID	Garmin to CCC	1

The type definition for the CCC_to_Garmin_ok_ack_text_msg_data_type is shown below. This data type is only supported on Garmins that report D602 as part of their protocol support data.

```
typedef struct /* D602 */
{
time_type origination_time;
uint8     id_size;
uint8     reserved[3]; /* set to 0 */
uint8     id[/* 16 characters */];
char      text_message[/* variable length, null-terminated string, 200 characters
max */];
} CCC_to_Garmin_ok_ack_text_msg_data_type;
```

The origination_time is the time that the text message was sent from the CCC. The id_size determines the number of characters used in the ID member. An id size of 0 indicates that there is no message id. The ID member is an array of 8-bit integers that could represent any type of data.

The type definition for the text_msg_ack_data_type is shown below. This data type is only supported on Garmins that report D602 as part of their protocol support data.

```
typedef struct /* D602 */
{
time_type origination_time;
uint8     id_size;
uint8     reserved[3]; /* set to 0 */ uint8      id[/* 16 characters */]; uint32
   msg_ack_type
} text_msg_ack_data_type;
```

The origination_time is the time that the text message was acknowledged on the Garmin. The id_size and ID will match the id_size and ID of the applicable text message. The msg_ack_type will depend on the type of text message that is being acknowledged. The table below defines the different values for msg_ack_type.

Value (Decimal)	Acknowledgment Type
0	Simple Okay Acknowledgement
1	Yes Acknowledgment
2	No Acknowledgment

5.4.1.8 CCC to Garmin Yes/No Confirmation Text Message Protocol

This text message protocol is used to send a Yes/No confirmation text message from the CCC to Garmin. When the Garmin receives this message, it displays the text message to the user. It also displays two buttons (Yes and No). The user is required to press one of the two buttons after reading the text message. Once the user presses one of the two buttons, the Garmin will send an acknowledgement message to the CCC. This protocol is only supported on Garmins that report A602 as part of their protocol support data.

The packet sequence for the CCC to Garmin Yes/No confirmation text message is shown below:

Fleet Management Packet Data Type	Fleet Management Packet ID	Direction	N
CCC_to_Garmin_yes_no_ack_text_msg_ data_type	0x0023 – CCC to Garmin Yes/No Confirmation Text Message Packet ID	CCC to Garmin	0
text_msg_ack_data_type	0x0020 – Text message Acknowledgment Packet ID	Garmin to CCC	1

The type definition for the CCC_to_Garmin_yes_no_ack_text_msg_data_type is shown below. This data type is only supported on Garmins that report D602 as part of their protocol support data.

```
typedef struct /* D602 */
{
time_type origination_time;
uint8     id_size;
uint8     reserved[3]; /* set to 0 */
uint8     id[/* 16 characters */];
char      text_message[/* variable length, null-terminated string, 200 characters max */];
} CCC_to_Garmin_yes_no_ack_text_msg_data_type;
```

The origination_time is the time that the text message was sent from the CCC. The id_size determines the number of characters used in the ID member. An ID size of 0 indicates that there is no message ID. The ID member is an array of 8-bit integers that could represent any type of data.

5.4.1.9 Garmin to CCC Open Text Message Protocol

This text message protocol is used to send a simple text message from the Garmin to the CCC. When the CCC receives this message, it sends a message receipt back to the Garmin. This protocol is only supported on Garmins that report A603 as part of their protocol support data. The packet sequence for the Garmin to CCC open text message is shown below:

Fleet Management Packet Data Type	Fleet Management Packet ID	Direction	N
Garmin_to_CCC_open_text_msg_data_type	0x0024 – Garmin to CCC Open Text Message Packet ID	Garmin to CCC	0
text_msg_receipt_data_type	0x0025 – Text Message Receipt Packet ID	CCC to Garmin	1

The type definition for Garmin_to_CCC_open_text_msg_data_type is shown below. This data type is only supported on Garmins that report D603 as part of their protocol support data.

```
typedef struct /* D603 */
{
time_type origination_time;
uint32    unique_id;
char      text_message[/* variable length, null-terminated string, 200 characters max */];
} Garmin_to_CCC_open_text_msg_data_type;
```

The origination_time is the time that the text message was sent from the Garmin terminal. The unique_id is the unsigned 32-bit unique identifier for the message.

The type definition for the text_msg_receipt_data_type is shown below. This data type is only supported on Garmins that report D603 as part of their protocol support data.

```
typedef struct /* D603 */
{
uint32    unique_id;
} text_msg_receipt_data_type;
```

The unique_id is the unsigned 32-bit unique identifier for the message that the Garmin terminal sent to the CCC.

5.5 Stop (Destination) Protocols

The Stop protocols are used to inform Garmin of a new destination. When the Garmin receives a Stop from the CCC, it displays the Stop to the user and also lets the user start navigate to the Stop location.

There are two protocols available to the CCC for sending Stops to a Garmin terminal: the A603 Stop protocol and the A602 Stop protocol. The A603 Stop protocol allows the Garmin to report the status of a Stop back to the CCC, while the A602 Stop protocol does not have the Stop status reporting capability. If a Garmin supports both the A603 and A602 Stop protocols, it is recommended that the CCC uses the A603 Stop protocol to send Stops to the Garmin.

5.5.1 *A603 Stop Protocol*

This protocol is used to send Stops or destinations from the CCC to the Garmin terminal. When the Garmin receives a Stop, it displays it to the user and lets the user start navigating to the Stop location. The Garmin reports Stop status (unread, read, active, etc.) for Stops it received using this protocol. This protocol is only supported on Garmins that report A603 as part of their protocol support data. The packet sequence for the A603 Stop protocol is shown below:

Fleet Management Packet ID	Fleet Management Packet Data Type	Direction	N
0x0101 – A603 Stop Protocol Packet ID	A603_stop_data_type	CCC to Garmin	0

The type definition for the A603_stop_data_type is shown below. This data type is only supported on Garmins that report D603 as part of their protocol support data.

```
typedef struct /* D603 */
{
time_type origination_time;
sc_position_type stop_position;
uint32    unique_id;
char      text[/* variable length, null-terminated string, 200 characters max */];
} A603_stop_data_type;
```

The origination_time is the time that the Stop was sent from the CCC. The stop_position is the location of the Stop. The unique_id contains a 32-bit unique identifier for the Stop. The text member contains the text that is displayed on Garmin's user interface for this Stop.

5.5.2 *A602 Stop Protocol*

This protocol is used to send Stops or destinations from the CCC to the Garmin. When the Garmin receives a Stop, it displays it to the user and lets the user start navigating to the Stop location. Garmin does not report Stop status (unread, read, active, etc.) for Stops it received using this protocol. This protocol is only supported on Garmins that report A602 as part of their protocol support data. The packet sequence for the A602 Stop protocol is shown below:

Fleet Management Packet ID	Fleet Management Packet Data Type	Direction	N
0x0100 – A602 Stop Protocol Packet ID	A602_stop_data_type	CCC to Garmin	0

The type definition for the A602_stop_data_type is shown below. This data type is only supported on Garmins that report D602 as part of their protocol support data.

```
typedef struct /* D602 */
{
time_type origination_time;
sc_position_type stop_position;
char      text[/* variable length, null-terminated string, 51 characters max */];
} A602_stop_data_type;
```

The origination_time is the time that the Stop was sent from the CCC. The stop_position is the location of the Stop. The text member contains the text that is displayed on the Garmin's user interface for this Stop.

5.5.3 *Stop Status Protocol*

This protocol is used by the CCC to request or change the status of a Stop on the Garmin. The protocol is also used by the Garmin to send the status of a Stop to the CCC whenever the status of a Stop changes on the Garmin. This protocol is only supported with Stops that are sent to the Garmin using the A603 Stop protocol.

This protocol is only supported on Garmins that report A603 as part of their protocol support data. The packet sequences for the Stop status protocol are shown below:

Fleet Management Packet Data Type	Fleet Management Packet ID	Direction	N
stop_status_data_type	0x0210 – Stop Status Request Packet ID	CCC to Garmin	0
stop_status_data_type	0x0211 – Stop Status Data Packet ID	Garmin to CCC	1
stop_status_receipt_data_type	0x0212 – Stop Status Receipt	CCC to Garmin	2
stop_status_data_type	0x0211 – Stop Status Data Packet ID	Garmin to CCC	0
stop_status_receipt_data_type	0x0212 – Stop Status Receipt	CCC to Garmin	1

The type definition for the stop_status_data_type is shown below. This data type is only supported on Garmins that report D603 as part of their protocol support data.

```
typedef struct /* D603 */
{
uint32    unique_id;
uint16    stop_status;
uint16    stop_index_in_list;
} stop_status_data_type;
```

The unique_id contains the 32-bit unique identifier for the Stop. The CCC should ignore any Stop status message with a unique_id of 0xFFFFFFFF. The stop_status depends on the current status of the Stop on the Garmin or the status to which the CCC would like to change the Stop. The stop_index_in_list can either represent the current position of the Stop in the Stop list (0, 1, 2…) if the message is going from the Garmin to the CCC or it represents the position the Stop should be moved to in the Stop list if the message is going from the CCC to the Garmin. The table below defines the stop_status and also explains how the value of the stop_status affects the contents of the stop_index_in_list.

Meaning	Value (Decimal)	Stop Status
This is the CCC requesting the status of a Stop from Garmin. The value of the stop_index_in_list should be set to 0xFFFF and is ignored by the Garmin.	0	Requesting Stop Status
This is the CCC telling the Garmin to mark a Stop as done. The value of the stop_index_in_list should be set to 0xFFFF and is ignored by the Garmin.	1	Mark Stop As Done

Meaning	Value (Decimal)	Stop Status
This is the CCC telling the Garmin to start navigating to a Stop. The value of the stop_index_in_list should be set to 0xFFFF and is ignored by the Garmin.	2	Activate Stop
This is the CCC telling the Garmin to delete a Stop from the list. The value of stop_index_in_list should be set to 0xFFFF and is ignored by the Garmin.	3	Delete Stop
This is the CCC telling the Garmin to move a Stop to a new position in the list. The value of stop_index_in_list should be set to the position the CCC would like to move the Stop to in the list.	4	Move Stop
This is the Garmin reporting the current status of a Stop as Active. The value of stop_index_in_list corresponds to the current position of the Stop in the list.	100	Stop status – Active
This is the Garmin reporting the current status of a Stop as Done. The value of stop_index_in_list will correspond to the current position of the Stop in the list.	101	Stop status –Done
This is the Garmin reporting the current status of a Stop as unread and inactive. The value of stop_index_in_list corresponds to the current position of the Stop in the list.	102	Stop status – Unread Inactive
This is the Garmin reporting the current status of a Stop as read and inactive. The value of stop_index_in_list corresponds to the current position of the Stop in the list.	103	Stop status – Read Inactive
This is the Garmin reporting the current status of a Stop as Deleted. The Garmin returns this status for any Stop that is not present in the Stop list. The value of stop_index_in_list is set to 0xFFFF and is ignored by the CCC.	104	Stop status – Deleted

The type definition for the stop_status_receipt_data_type is shown below. This data type is only supported on Garmins that report D603 as part of their protocol support data.

```
typedef struct /* D603 */
{
uint32    unique_id;
} stop_status_receipt_data_type;
```

The unique_id contains the 32-bit unique identifier for the Stop.

5.5.4 *Estimated Time of Arrival (ETA) Protocol*

This protocol is used by the CCC to request ETA and destination information from the Garmin. The Garmin also uses this protocol to send ETA and destination information to the CCC whenever the user starts navigating to a new destination. This protocol is only supported on Garmins that report A603 as part of their protocol support data. The packet sequences for the ETA protocol are shown below:

Fleet Management Packet Data Type	Fleet Management Packet ID	Direction	N
None	0x0200 – ETA Data Request Packet ID	CCC to Garmin	0
eta_data_type	0x0201 – ETA Data Packet ID	Garmin to CCC	1
eta_data_receipt_type	0x0202 – ETA Data Receipt Packet ID	CCC to Garmin	2
eta_data_type	0x0201 – ETA Data Packet ID	Garmin to CCC	0
eta_data_receipt_type	0x0202 – ETA Data Receipt Packet ID	CCC to Garmin	1

The type definition for the eta_data_type is shown below. This data type is only supported on Garmins that report D603 as part of their protocol support data.

```
typedef struct /* D603 */
{
uint32    unique_id;
time_type eta_time;
uint32    distance_to_destination;
sc_position_type position_of_destination;
} eta_data_type;
```

The unique_id is a 32-bit unsigned value that uniquely identifies the ETA message sent to the CCC. The eta_time is the time that Garmin expects to arrive at the currently active destination. If the eta_time is set to 0xFFFFFFFF, then the Garmin does not have a destination active. The distance_to_destination is the distance in meters from the Garmin to the currently active destination. If the distance_to_destination is set to 0xFFFFFFFF, then the Garmin does not have a destination active. The position_of_destination is the location of the currently active destination on Garmin.

The type definition for the eta_data_receipt_type is shown below. This data type is only supported on Garmins that report D603 as part of their protocol support data.

```
typedef struct /* D603 */
{
uint32    unique_id;
} eta_data_receipt_type;
```

The unique_id is a 32-bit unsigned value that uniquely identifies the ETA message sent to the CCC.

5.5.5 *Auto-Arrival at Stop Protocol*

This protocol is used by the CCC to change the auto-arrival criteria on the Garmin. The auto-arrival feature is used on the Garmin to automatically detect that the user has arrived at a Stop and then to prompt the user if he/she would like to mark the Stop as done and start navigating to the next Stop in the list. Once the CCC sends the auto-arrival at Stop protocol to the Garmin, the setting is permanent on the Garmin until the CCC changes it.

This protocol is only supported on Garmins that report A603 as part of their protocol support data. The packet sequence for the Auto-Arrival at Stop protocol is shown below:

Fleet Management Packet Data Type	Fleet Management Packet ID	Direction	N
auto_arrival_data_type	0x0220 – Auto-Arrival Data Packet ID	CCC to Garmin	0

The type definition for the auto_arrival_data_type is shown below. This data type is only supported on Garmins that report D603 as part of their protocol support data.

```
typedef struct /* D603 */
{
uint32    stop_time;
uint32    stop_distance;
} auto_arrival_data_type;
```

The stop_time value is the time in seconds for the period that the Garmin should be stopped, when close to the destination, before the auto-arrival feature is activated. The default for stop_time on Garmin is 30 seconds. To disable the auto-arrival stop time, set stop_time to 0xFFFFFFFF. The stop_distance is distance in meters for the distance that the Garmin has to be to the destination before the auto-arrival feature is activated. The default for stop_distance on Garmin is 100 meters. To disable the auto-arrival stop distance, set stop_distance to 0xFFFFFFFF. To totally disable the auto-arrival feature, set both stop_time and stop_distance to 0xFFFFFFFF.

5.5.6 *Data Deletion Protocol*

This protocol is used by the CCC to delete data on the Garmin. This protocol is only supported on Garmins that report A603 as part of their protocol support data. The packet sequence for the Data Deletion protocol is shown below:

Fleet Management Packet Data Type	Fleet Management Packet ID	Direction	N
data_deletion_data_type	0x0230 – Data Deletion Packet ID	CCC to Garmin	0

The type definition for the data_deletion_data_type is shown below. This data type is only supported on Garmins that report D603 as part of their protocol support data.

```
typedef struct /* D603 */
{
uint32    data_type;
} data_deletion_data_type;
```

The value for the data_type corresponds to the type of data to be deleted on the Garmin. The table below defines the values for data_type.

Value (Decimal)	Meaning
0	Deletes all Stops on the Garmin
1	Deletes all messages on the Garmin

5.6 Other Relevant Garmin Protocols

All the protocols described in this section are Garmin protocols that are supported on all Garmin devices that support fleet management. The protocols are not related to the fleet management, but can prove to be very useful to create a complete fleet management system design.

5.6.1 *Command Protocol*

This section describes a simple protocol used in commanding the Garmin or CCC to do something (e.g. send position data). For a list of command IDs relevant to this document, please refer to the Application Note. The link layer packet for the command protocol should be represented as follows:

Notes	Byte Description	Byte Number
16 (decimal)	Data Link Escape	0
10 (decimal)	Packet ID	1
2	Size of Packet Data	2
See Application Note	Command ID	3-4
2's complement of the sum of all bytes from byte 1 to byte 4	Checksum	5
16 (decimal)	Data Link Escape	6
3 (decimal)	End of Text	7

5.6.2 *Unit ID/ESN Protocol*

Packet Data Type	Packet/Command ID	Direction	N
No data (command)	14 – Request Unit ID Command ID	CCC to Garmin	0
unit_id_data_type	38 – Unit ID Packet ID	Garmin to CCC	1

This protocol is used to extract Garmin's unit ID (or electronic serial number). The packet sequence for this protocol is shown below.

The type definition for the unit_id_data_type is shown below.

```
typedef struct
{
uint32    unit_id;
uint32    other_stuff[ /* variable length */ ];
} unit_id_data_type;
```

The unit_id is the first 32-bits of the data type. Some Garmins can append additional information that is used by Garmin in the manufacturing process. This data should be ignored.