LUIS Gen2 DLL API

Loadbox User Interface System Gen2 DLL API

Version 1.1

3037 W St. Rd. 256
Austin, In 47102
www.gartechenterprises.com

Dated: 07-08-2013

Revision History

This revision history only shows major modifications between release versions.

Date	Author	Filename		Comments
07-08-2013	K.Stephenson	The LUIS Client DLL Documentation Rev1_1.docx	-	Initial Release

Table of Contents

Introduction

The LUIS Client DLL will allow a user to interface directly to the LUIS Hardware via a PC windows service. When the LUIS Gen2 software gets installed, serveral Windows services also get installed and start running. Those services provide the communication link from any client application to the LUIS hardware. The client only has to create an instance of the LUIS Gen2 DLL and instruct it to connect to the appropriate service which will be called "Server". The DLL sends the simple text commands to the connected Server. Those simple text commands are converted in the multi-client server to the appropriate format, protocol, and communication link which are then sent to the hardware.
A typical topology of the communications is depicted in Figure 1. In this figure, the LUIS PC software acts as a client and is shown connected to three Servers(LUISGen2_#1, WaveMaker_#1, PeakAdapter_#1). For each client/server connection a separate DLL must be instantiated. In this example the LUIS software created three instances of the client DLL.

LUIS Gen2 Client/Server Connectivity

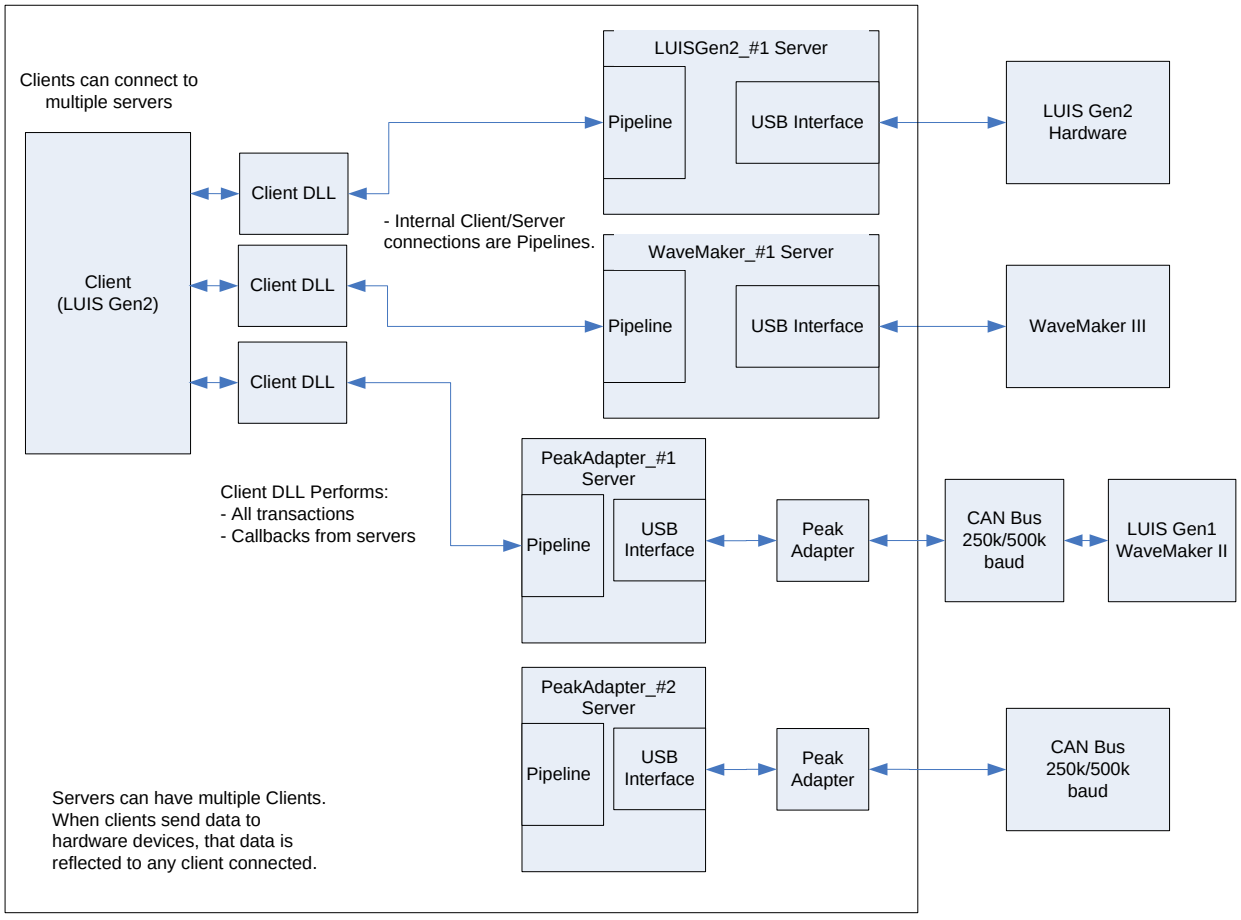

Figure 1. Server Communication Connections

Programming to the DLL

In order to use this C# DLL the user will need to create an instance of the LUISClient and set the following items:

- Name: The name of the connection.
- Pipename: The pipename that has been given to the server in the xml file. (Default LUISGen2_#1)
- Type: 0=Pipe,1=TCP,2=HTTP (Default = 0)

There are three events that can be hooked into to receive information from the server. These events are:

- ClientError: Event for any errors.
- Disconnected: Triggered when the client is disconnected.
- MessageReceived: Event when messages are pushed to the client.

Once all the values have been set the user must call StartService() function to initiate the connection to the server.
To send a message to the server the user must use the SendMsg(string msg) function.

For Microsoft Visual Studio Development:
A reference must be made in order to use the DLL. See figure 2.

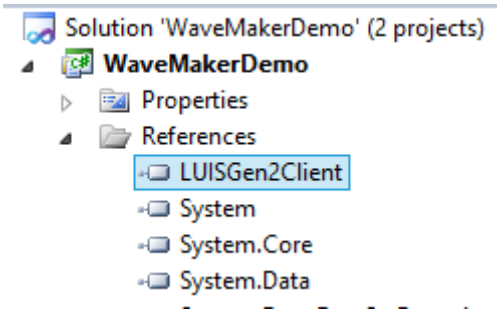

Figure 2. LUISGen2Client Reference in Visual Studio.

Declare the LUIS client(C# Example):

```
LUISClient _client;
```

Connect and Start the Client:

```
// Create new object and connect events
_client = new LUISClient();
_client.ClientError += new LUISClient.ErrorHandler(_client_ClientError);
_client.Disconnected += new EventHandler(_client_Disconnected);
_client.MessageReceived += new
LUISClient.MessageReceivedHandler(_client_MessageReceived);

// Set the connection name to the pipe and make the connection
_client.ConnectionName = "WaveMaker_#1";
_client.StartService();
```

To send a message to the hardware a simple text message can be sent:

```
_client.SendMsg("Control:WaveMaker,ResetAll");
```

The above message will reset the WaveMaker unit.

When all operations are complete and the client is going to close a StopService() call should be sent to properly disconnect the client from the server:

```
_client.StopService();
```

There are four Servers that can be connected to. Their Connection Names are:

"LUISGen2_#1":	Commands Main, Analogs, Switch, Loads
"WaveMaker_#1":	Commands WaveMaker, Closed Loop
"PeakAdapter_#1":	Commands Gen1 or any CAN message
"PeakAdapter_#2":	Commands Gen1 or any CAN message

If the client application wants to connect to all four of the above servers then there has to be four instances of LUISClient and each one connecting to their own server.

Plugin Names

Each command must target a specific plugin to be executed properly. The following names must be used exactly as shown.

LUIS Gen1 System

- "Parent Main (LUISGen1)"
- "Parent Analog Output (LUISGen1)"
- "Parent Switch (LUISGen1)"
- "Child 1 Analog Output (LUISGen1)"
- "Child 1 Main (LUIS Gen1)"
- "Child 1 Switch (LUIS Gen1)"
- "Child 2 Analog Output (LUISGen1)"
- "Child 2 Main (LUIS Gen1)"
- "Child 2 Switch (LUIS Gen1)"
- "Sidecar Analog Output (LUISGen1)"
- "Sidecar Main (LUIS Gen1)"
- "Sidecar Switch (LUIS Gen1)"
- "Wavemaker (LUISGen1)"

LUIS Gen2 System

- "Main Module"
- "Analog Output"
- "Switch"
- "MainModuleJ1939PortAPlugin"
- "Resistive Load"
- "WaveMaker"

FMET System

- "Control Board (FMET)"
- "Relay Board 1 (FMET)"
- "Relay Board 2 (FMET)"
- "Relay Board 3 (FMET)"
- "Relay Board 4 (FMET)"
- "Relay Board 5 (FMET)"
- "Relay Board 6 (FMET)"

Other Plugin Names

- "PeakAdapterPlugin"

LUIS Gen1 Plugin Commands

Parent Main (LUISGen1)

Child 1 Main (LUIS Gen1)

Child 2 Main (LUIS Gen1)

Sidecar Main (LUIS Gen1)

Version

Is the reponse back from the hardware when a GetAllVersionData has been requested. Each piece of hardware will send its own unique message depicting the version/firmware it is.

Format = "Version:[Plugin Name],[Part Number],[Serial Number],[Firmware Ver]"

Response from Hardware:

"Version:Parent Main (LUIS Gen1),N/A,Main,1.1"

Firmware Version = 1.1

PowerUp

Is fired back into the client when the server detects the PowerUp message from the Main LUIS Gen1 unit on the CAN bus.

Format = "PowerUp:Parent Main (LUIS Gen1)"

Data

The command is only received by the client when the status message on the CAN bus has been received.

Format = "Data:Parent Main (LUISGen1),[Lamp Status Number],[Value]"

Response from Hardware:

"Data:Parent Main (LUIS Gen1),LampStatus#1,1"

Lamp #1 is turned ON

Lamp Status Numbers:

"LampStatus#1"

…

"LampStatus#5"

GetAllVersionData

Requests the version of the firmware from the device. The Version command will get fired back into the client.

Format = "GetInfoNoWait:Parent Main (LUIS Gen1),GetAllVersionData"

Reset

Resets the LUIS Gen1 system.

Format = “Control:Parent Main (LUISGen1),Reset”

HardReset

Performs a hard reboot of the LUIS Gen1 system, useful for ROM booting.

Format = “Control:Parent Main (LUISGen1),HardReset”

Parent Analog Output (LUISGen1)
Child 1 Analog Output (LUISGen1)
Child 2 Analog Output (LUISGen1)
Sidecar Analog Output (LUISGen1)

Data

Format = “Data:Analog Output,[Analog Number],[Value]”

Analog Numbers =

“Analog Output#1”

. . . .

“Analog Output#32”

Value = hardware counts (0-16383 for 14 bit resolution)

Example Command =

“Data:Parent Analog Output (LUISGen1),Analog Output#26,1219”

Sets the Analog Output #26 value to 1219 counts.

Note:

Sidecar Analog Output only has 12 channels and the channels 9-12 are only 8 bit resolution.

Parent Switch (LUISGen1)
Child 1 Switch (LUIS Gen1)
Child 1 Switch (LUIS Gen1)
Sidecar Switch (LUIS Gen1)

Data

Format = “Data: [Plugin Name],[Switch Number],[Value]”

Switch Numbers =

“Switch#1”

. . . .

“Switch#32”

Value = 0 is OFF 1 is ON

Example Command =

“Data:Parent Switch (LUISGen1),Switch#32,1”

Sets the Switch#32 to ON.

Note:

Sidecar only has 8 switches.

Reset

Resets the LUIS Gen1 switches.

Format = “Control: Parent Switch (LUISGen1),Reset”

Wavemaker (LUISGen1)

Version

Is the reponse back from the hardware when a GetAllVersionData has been requested. Each piece of hardware will send its own unique message depicting the version/firmware it is.

Format = “Version:[Plugin Name],[Part Number],[Serial Number],[Firmware Ver]”

Response from Hardware:

“Version:Wavemaker (LUIS Gen1),N/A,WaveMaker,1.1”

Firmware Version = 1.1

PowerUp

Is fired back into the client when the server detects the PowerUp message from the Main LUIS Gen1 unit on the CAN bus.

Format = “PowerUp: Wavemaker (LUIS Gen1)”

GetAllVersionData

Requests the version of the firmware from the device. The Version command will get fired back into the client.

Format = “GetInfoNoWait: Wavemaker (LUIS Gen1),GetAllVersionData”

Reset

Resets the Wavemaker in the LUIS Gen1 system.

Format = “Control: Wavemaker (LUISGen1),Reset”

HardReset

Performs a hard reboot of the Wavemaker in the LUIS Gen1 system, useful for ROM booting.

Format = “Control: Wavemaker (LUISGen1),HardReset”

Data

Format = “Data: [Plugin Name],[Channel Number],[Value]”

Channel Numbers =

“Channel#1”

. . . .

“Channel#8”

Value = 0 to 16,777,215

Example Command =

"Data:Wavemaker (LUISGen1),Channel#1,141"

Sets the Channel#1 to a frequency value of 141 based on how the channel is setup.

ChannelConfig

The channel config command will setup the instructed channel for a specific operation.

Format = "ChannelConfig:Wavemaker (LUISGen1),[Channel Number],[Offset],[Teeth/Rev],[Hall/VR],[Freq/RPM],[Cycles/rev],[Sync],[Card Type],[Waveform Name]"

Channel Numbers =

"Channel#1"

…

"Channel#8"

Offset = 0 – 65535

Teeth/Rev = 0 – 65535

Hall/VR =

"Hall"

"VR"

Freq/RPM =

"Freq"

"RPM"

Cycles/Rev = 0 – 15

Sync =

"Master"

"Slave"

Card Type =

"Arbitary"

"Digital"

"DigitalSim"

Waveform Name = The name of the waveform that is being sent to Wavemaker.

Example Command =

"ChannelConfig:Wavemaker (LUISGen1),Channel#1,0,0,0,Hall,RPM,2,Master,2,ESS 60-2"

Sets channel number 1 to be an arbitrary card that has 2 cycles per rev and instructs it to load waveform "60-2" into that channel. It is also set as the Master and will control any Slave channels.

ClosedLoopData

Runs the simple closed loop model in the WaveMaker. Standard J1939 messages must be broadcast in order for this to run.

Format = “ClosedLoopData:Wavemaker (LUISGen1),[Channel Number],[Command],[Value]”

Channel = The channel that is being controlled by the closed loop system.

“Channel#1”

. . .

“Channel#8”

Command =

“Gain” = Adjust the gain for the model (0-65535)

“PercentLoad” = Adjust the load on the engine (-125% to +125%)

“ClosedLoop” = Run the system in a Closed loop mode

“OpenLoop” = Run the system in an Open loop mode

“Start” = Start the engine

“Reset” = Reset the model

Value = Used for the “Gain” and “PercentLoad” commands.

Example Command =

“ClosedLoopData: Wavemaker (LUISGen1),Channel#1,PercentLoad,20”

Sets the Percent Load on the engine to 20%

LUIS Gen2 Plugin Commands

Main Module

Data

Sets the hardware value to the value in the command. The destination name comes from the Plugin.

Format = “Data:Main Module,[Main Module Number],[Value]”

Main Module Numbers =

“VBATT Relay#1”

“Switched VBATT Relay#1”

. . .

“Switched VBATT Relay#4”

Value = 0 or 1

Example Command =

“Data:Main Module,VBATT Relay#1,1”

Turns on the Main VBATT relay

GetData

Requests the current status of the hardware value.

Format = “GetData:Main Module,[Main Module Number]”

Main Module Numbers =

“VBATT Relay#1”

“Switched VBATT Relay#1”

. . .

“Switched VBATT Relay#4”

Example Command =

“Data:Main Module,VBATT Relay#1”

Gets the value of Main VBATT relay#1

Control

Format = “Control: Main Module,[Command]

Command =

“Reset”

“ResetAll”

Example Command =

“Control: Main Module,ResetAll”

Resets all of the modules

Analog Output

Data

Sets the hardware value to the value in the command. The destination name comes from the Plugin.
Format = “Data:Analog Output,[Analog Number],[Value]”

Analog Numbers =
“Analog Output#1”
. . . .
“Analog Output#128”
Value = hardware counts (0-65535 for 16 bit resolution)

Example Command =
“Data:Analog Output,AnalogOutput#1,1000”
Sets the Analog Output module#1 value to 1000 counts.

GetData

Requests the current status of the hardware value.
Format = “GetData:Analog Module,[Analog Number]

Analog Number =
“Analog Output#1”
. . . .
“Analog Output#128”

Example Command =
“GetData:Analog Module,Analog Output#1”
Gets the value of the first Analog channel in counts.

Switch

Data

Sets the hardware value to the value in the command. The destination name comes from the Plugin.

Format = ”Data:Switch,[Switch Number],[Value]”

Switch Numbers =

“Switch #1”

. . .

“Switch #161”

Value = 0 or 1

Example Command:

“Data: Switch, Switch #1,1”

Turns ON the first Switch of the Switch modules

GetData

Requests the current status of the hardware value.

Format = ”Data: Switch,[Switch Number]”

Switch Numbers =

“Switch #1”

. . .

“Switch #161”

Example Command =

“Data: Switch, Switch #1”

Get the value of Switch #1

MainModuleJ1939PortAPlugin

Data

Sets the hardware value to the value in the command. The destination name comes from the Plugin.

Format = ”Data: MainModuleJ1939PortAPlugin,[PGN],[Byte 1-7]”

PGN = The decimal value of the Parameter Group Number that is being changed.

Byte1-7 = The 8 data bytes for the PGN

Example Command =

“Data: MainModuleJ1939PortAPlugin, 418316794,8,1,255,3,100,0,255,255”

Sets the 8 data bytes for 0x18EF01FA pgn

CANConfig

Setup a CAN message in the hardwares transmit buffer.

Format = “CANConfig: MainModuleJ1939PortAPlugin,[PGN],[Rate],[#Bytes],[id]

PGN = Parameter Group Number in decimal format

Rate = Transmission rate in milliseconds

#Bytes = The number of Bytes in the CAN message, maximum is 8

Id = The ID of the CAN module

1 = Main Module External CAN bus

Example Command =

“CANConfig: MainModuleJ1939PortAPlugin, 418316794,20,8,1

Transmit 0x18EF01FA message every 20mS on Main CAN bus

ClearList

Clears the list of CAN messages that are being transmitted.

Format = “ClearList: MainModuleJ1939PortAPlugin”

StartBroadcast

Starts the CAN message table broadcasting if messages are created.

Format = “StartBroadcast: MainModuleJ1939PortAPlugin”

SetBusSpeed

Sets the bus speed for the transmit buffer in the LUIS Gen2 hardware.

Format = “SetBusSpeed: MainModuleJ1939PortAPlugin,[Bus Speed]

Bus Speed =

"250k"
"500k"

Example:

"SetBusSpeed: MainModuleJ1939PortAPlugin,250k"
Sets the bus speed for 250k operation.

Resistive Load

GetDataBroadcast

Requests the status from Resistive Load modules.

Format = “GetDataBroadcast:Resistive Load,[Resistive Load Bank]”

Resistive Load Bank =

“Resisitve Load#0” = Load status for loads 1-36

“Resistive Load#36” = Load status for loads 37-72

Example =

“GetDataBroadcast:Resistive Load,Resistive Load#0”

DataMux

The response from hardware that contains the status of the loads.

Format = “DataMux:ResistiveLoadAPlugin,[Resistive Load Bank],[loads 1-8],[loads 9-16],[loads 17-24],[loads 25-32],[loads 33-36]”

Resistive Load Bank =

“0” = Load status for loads 1-36

“35” = Load status for loads 37-72

Example Response:

“DataMux:ResistiveLoadAPlugin,0,0,0,0,0,0”

The bit encoding is from low bit to high bit. Least significant bit is the lowest resistive load number in that byte.

WaveMaker

Data

Sets the hardware value to the value in the command. The destination name comes from the Plugin.

Format = ”Data:WaveMaker,[Channel],[Value]”

Channel =

“Arbitrary_CH#1”

. . .

“Arbitrary_CH#8”

“Digital_CH#1”

. . .

“Digital_CH#10”

Value = Hz or RPM depending on how the channel is setup

Example Command =

“Data:WaveMaker,Arbitrary_CH#1,1000”

Sets the first Arb channel to 1000 RPM

GetData

Requests the current status of the hardware value.

Format = ”Data:WaveMaker,[Channel]”

Channel =

“Arbitrary_CH#1”

. . .

“Arbitrary_CH#8”

“Digital_CH#1”

. . .

“Digital_CH#10”

Example Command =

“Data:WaveMaker,Arbitrary_CH#1”

Get the value of Arbitrary Channel #1

ChannelConfig

Setup a channel for the WaveMaker.

Format =

“ChannelConfig:WaveMaker,[Channel],[Offset],[Teeth],[PWM],[Hall/VR],[Freq/RPM],[Cycles/Rev],[Sync],[External Sync],[Master Channel]”

Channel =

"Arbitrary_CH#1"
. . .
"Arbitrary_CH#8"
"Digital_CH#1"
. . .
"Digital_CH#10"

Offset = The number of data points to offset data in relation to other waveforms.
Teeth = The number of teeth per revolution of a flywheel, used to make the RPM calculations.
PWM = The heartbeat fequency if the channel is setup to be used as a PWM instead of frequency. A non-zero value makes the channel configured to be PWM based.
Hall/VR = The level that the digital output has.
"VR" = variable reluctance(+5v to -5v)
"Hall" = Hall effect (+5v)
Freq/RPM = The type of values that will be sent using the "Data" command.
"RPM" = "Data" commands are in RPM values.
"Freq" = "Data" commands are in frequency values.
Cycles/Rev = The number of cycles that the bit map data represents. This value is needed for a correct RPM calculation on an Arbitrary channel.
Sync = Syncs the Aribitrary channel to the a Master channel. This is needed to lock channels together such as Engine speed and Engine position.
"Slave" = Set this channel to be a slave to another channel so frequency changes are done by the Master channel automatically.
"Master"
External Sync = Syncronize this channel with an external input.
"1" = External Sync enable
"0" = External Sync disable
Master Channel = The channel that controls this channel if it is a slave channel.
"Arbitrary_CH#1"
. . .
"Arbitrary_CH#8"
"Digital_CH#1"
. . .
"Digital_CH#10"
Null entry denotes that this is a Master Channel

ChannelDataLoad

Send the arbitrary waveform data to the card.
Format = "ChannelDataLoad:WaveMaker,[Channel Listing],[Name],[Length],[Data]"
Channel Listing = All the channels that this data gets loaded into seperated by a semicolon.

"Arbitrary_CH#1"
. . .
"Arbitrary_CH#8"
"Digital_CH#1"
. . .
"Digital_CH#10"

Name =The name of the waveform 15 characters or less.
Length = The length of the data bit map.
Data = Arbitrary data values in mV resolution.

Example Command =

"ChannelDataLoad:WaveMaker,Arbitrary_CH#1;Arbitrary_CH#2,Test Waveform,6,5000,0,5000,0,5000"

Loads arbitrary channels 1 and 2 with the same Test Waveform data which is 6 data points in length.

ClosedLoopData

Runs the simple closed loop model in the WaveMaker. Standard J1939 messages must be broadcast in order for this to run.

Format = "ClosedLoopData:WaveMaker,[Channel],[Command],[Value]"

Channel = The channel that is being controlled by the closed loop system.

"Arbitrary_CH#1"
. . .
"Arbitrary_CH#8"
"Digital_CH#1"
. . .
"Digital_CH#10"

Command =

"Gain" = Adjust the gain for the model (0-65535)
"PercentLoad" = Adjust the load on the engine (-125% to +125%)
"ClosedLoop" = Run the system in a Closed loop mode
"OpenLoop" = Run the system in an Open loop mode
"Start" = Start the engine
"Reset" = Reset the model

Value = Used for the "Gain" and "PercentLoad" commands.

Example Command =

"ClosedLoopData:WaveMaker,Arbitrary_CH#1,PercentLoad,20"

Sets the Percent Load on the engine to 20%

Common LUIS Gen2 Commands

Version

Is the reponse back from the hardware when a GetAllVersionData has been requested. Each piece of hardware will send its own unique message depicting the version/firmware it is.

Format = “Version:[Plugin Name],[Part Number],[Serial Number],[Firmware Ver]”

Response from Hardware:

“Version:Analog Output,102,1001,1.1”

Part Number = 102, Serial Number = 1001, Firmware Version = 1.1

Reset

Format = “Control: WaveMaker,[Command]

Command =

“Reset”

“ResetAll”

Example Command =

“Control: WaveMaker,Reset”

Resets the WaveMaker module

DownloadFirmware

Downloads a specified firmware file.

Format = “DownloadFirmware:[Module Name],[Part Number],[Data]

Module Name =

“Main Module”

“Analog Output”

“Relay”

“Resistive Load”

“WaveMaker”

Part Number =

Main Module = “16”

Analog Module = “48”

Relay = “32”

Resistive Load = “64”

WaveMaker = “96”

Data = Data bytes seperated by a “,”

GetInfo

Requests information from the modules.

Format = “GetInfo:[Module Name],[Command]

Module Name =

“Main Module”

“Analog Output”

“Relay”

“Resistive Load”

“WaveMaker”

Command =

“GetAllVersionData”

Requests all like modules to return version information

Example =

“GetInfo:Main Module,GetAllVersionData”

Returns the Main Modules version information

FMET System Plugin Commands

Control Board (FMET)

SetMaxCurrent

Sets the maximum amount of current that the system will tolerate before shutting down the fault relays.

Format = “Data:FMET Control Board (FMET),SetMaxCurrent,[Current]

Current = 0-20A scaled value by 0.125 ([Current]/0.125)

SetFault

Set the type of fault to the proper location.

Format = “FMETControl:FMET Control Board (FMET),SetFault,[Fault Type],[Location]”

Fault Type =

“Open”

“VBATT”

“GND”

“Fault 1”

“Fault 2”

Location =

“ECM”

“Both”

“Harness”

Example =

“FMETControl:FMET Control Board (FMET),SetFault,VBATT,ECM”

Shorts the connections that are connected to the ECM rail to VBATT.

ClearFault

Clear the faults.

Format = “FMETControl:FMET Control Board (FMET),ClearFault”

Relay Board 1 (FMET)
Relay Board 2 (FMET)
Relay Board 3 (FMET)
Relay Board 4 (FMET)
Relay Board 5 (FMET)
Relay Board 6 (FMET)

Data
Turn the relays on and off.

Format = “Data:[Relay Board Plugin],[Relay Number],[Value]”

Relay Number =

“Relay#1”

…

“Relay#180”

Value = 0 is OFF 1 is ON

Example =

“Data: Relay Board 1 (FMET),Relay#1,1”

Will turn Relay #1 ON when the fault gets applied.

SetFault
Turn on all relays that are supposed to be faulted.

Example =

“FMETControl:Relay Board 1 (FMET),SetFault”

ClearFault
Clear the faults.

Example =

“FMETControl: Relay Board 1 (FMET),ClearFault”

CurrentFeedbackBoard
This message is a response from the FMET system that contains the highest current drain value on that board.

Format = “Data:[Relay Board Plugin],[Relay Board],[Value]”

Relay Board =

"CurrentFeedbackBoard#1"

…

"CurrentFeedbackBoard#6"

Value = 0-20A scaled value by 0.125 ([Current]/0.125)

RelayErrorBoard

This message is a response from the FMET system that contains any errors that occur while setting the relays.

Format = "Data:[Relay Board Plugin],[Relay Board],[Value]"

Relay Board =

"RelayErrorBoard#1"

…

"RelayErrorBoard #6"

Value = 0 is no error and 1 is error.

Common FMET System Commands

Version

Is the reponse back from the hardware when a GetAllVersionData has been requested. Each piece of hardware will send its own unique message depicting the version/firmware it is.

Format = “Version:[Plugin Name],[Part Number],[Serial Number],[Firmware Ver]”

Response from Hardware:

“Version:Relay Board 1 (FMET),N/A,RelayBoard1,1.1”

Firmware Version = 1.1

PowerUp

Is fired back into the client when the server detects the PowerUp message from the FMET system unit on the CAN bus.

Format = “PowerUp:Relay Board 1 (FMET)”

GetAllVersionData

Requests the version of the firmware from the device. The Version command will get fired back into the client.

Format = “GetInfoNoWait: Relay Board 1 (FMET),GetAllVersionData”

Reset

Resets the system.

Format = “Control: Relay Board 1 (FMET),Reset”

Peak Adapter Plugin Commands

With the use of a Peak Datalink Adapter, the user can subscribe to specific pieces of J1939 data and have that data passed back when the messages arrive.

AddPGN

Adds a PGN to the list to subscribe to.

Format =
"PGNData:PeakAdapterPlugin,AddPGN,[PGN],[Parameter],[StartBit],[Length],[Multiplier],[Offset]"

PGN = The PGN number in decimal
Parameter = The first parameter name
StartBit = The start bit location of the parameter
Length = The length in bits of the parameter
Multiplier = The multiplier of the parameter
Offset = The offset of the parameter

Additional Parameters can be defined after the PGN as long as they are fully described.

RemovePGN

Removes a PGN in the subscribed listing

Format = "PGNData:PeakAdapterPlugin,RemovePGN,[PGN]"

PGN = The PGN number in decimal

ClearList

Clears the PGN subscribed listing

Format = "PGNData:PeakAdapterPlugin,ClearList"

Data

When the subscribed parameter from the PGN is received from the Peak adapter, it will call back the client.

Format = "Data:PeakAdapterPlugin,[Parameter Name],[Cooked value]

Parameter = The parameter name that is subscribed to
Cooked Value = The value of the parameter properly scaled