LUIS Client DLL API

Loadbox User Interface System Client DLL API

Version 2.1

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

Dated: 04-02-2022

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
01-18-2022	B.Strothman	The LUIS Client DLL Documentation Rev2.docx	- Several corrections and additions for Gen2 commands - Added Gen3 Commands
04-02-2022	B.Strothman	The LUIS Client DLL Documentation Rev2_1.docx	- Added Selector commands

Table of Contents

Figures 7

Introduction 8

Programming to the DLL 10

Sending Commands 12

Client Events 12

Plugin Names 13

  LUIS Gen2 and Gen3 Systems 13

  LUIS Gen3 Systems Only 13

  LUIS Gen1 System 13

  FMET System 14

  Peak Adapter 14

  Common LUIS Gen2 and Gen3 Commands 15

    Reset 15

    DownloadFirmware 15

    GetInfo 15

    Version 16

LUIS Gen2 and Gen3 Plugin Commands 16

  Main Module 16

    Data 16

    GetData 16

    Control 17

    Action 17

  Analog Output Module 20

    Data 20

    GetData 20

  Switch Module 21

    Data 21

    GetData 21

  Resistive Load 21

GetDataBroadcast 21
DataMux (Response from GetDataBroadcast Command) 22
Rheostat Module (LUIS Gen3 Only) 22
Data 22
GetData 22
Selector Module (LUIS Gen3 Only) 23
Data 23
GetData 23
MainModuleJ1939PortAPlugin 23
MainModuleJ1939PortBPlugin (LUIS Gen3 Only) 23
MainModuleJ1939PortCPlugin (LUIS Gen3 Only) 23
Data 23
CANConfig 24
ClearList 24
StartBroadcast 24
StopBroadcast 24
SetBusSpeed (LUIS Gen2 Only) 25
GetBusSpeed (LUIS Gen3 Only) 25
GetBusSpeed Response(LUIS Gen3 Only) 25
WaveMaker 26
Data 26
GetData 26
ChannelConfig 26
ChannelDataLoad 28
ClosedLoopData 29
LUIS Gen1 Plugin Commands 31
Parent Main (LUISGen1) 31
Child 1 Main (LUIS Gen1) 31
Child 2 Main (LUIS Gen1) 31
Sidecar Main (LUIS Gen1) 31

Version 31
PowerUp 32
Data 32
GetAllVersionData 32
Reset 32
HardReset 32
Parent Analog Output (LUISGen1) 33
Child 1 Analog Output (LUISGen1) 33
Child 2 Analog Output (LUISGen1) 33
Sidecar Analog Output (LUISGen1) 33
Data 33
Parent Switch (LUISGen1) 34
Child 1 Switch (LUIS Gen1) 34
Child 1 Switch (LUIS Gen1) 34
Sidecar Switch (LUIS Gen1) 34
Data 34
Reset 34
Wavemaker (LUISGen1) 35
Version 35
PowerUp 35
GetAllVersionData 35
Reset 35
HardReset 35
Data 35
ChannelConfig 36
ClosedLoopData 37
FMET System Plugin Commands 37
Control Board (FMET) 37
SetMaxCurrent 37
SetFault 37

ClearFault 38
Relay Board 1 (FMET) 38
Relay Board 2 (FMET) 38
Relay Board 3 (FMET) 38
Relay Board 4 (FMET) 38
Relay Board 5 (FMET) 38
Relay Board 6 (FMET) 38
Data 38
SetFault 39
ClearFault 39
CurrentFeedbackBoard 39
RelayErrorBoard 39
Common FMET System Commands 40
Version 40
PowerUp 40
GetAllVersionData 40
Reset 40
Peak Adapter Plugin Commands 41
AddPGN 41
RemovePGN 41
ClearList 41
Data 41

Figures

Figure 1: LUIS Gen2 Server Connections........8
Figure 2: LUIS Gen3 Server Connections........9
Figure 3: LUIS Plugin Usage........9
Figure 4: LUISGen2Client Reference in Visual Studio. 10
Figure 5: Creating and Connecting to the LUISClient object Example........11
Figure 6: Command Path and Conversion 11
Figure 7: Reflected Commands to Clients........11
Figure 8:Connecting to Events 13
Figure 9:Disconnect from Events 13
Figure 10: Toggle Switch Encodings........17

Introduction

The LUIS Client DLL will allow a user to interface directly to the LUIS Hardware via a PC windows service. When the LUIS PC software is installed, serveral Windows services that support the LUIS software will 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 via USB or CAN interfaces.

A typical topology of the communications is depicted in Figure 1 for LUIS Gen2. 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 LUISClient DLL. Each instance of the LUISClient DLL is used to communicate with a server (Windows Service) that they have been connected to.

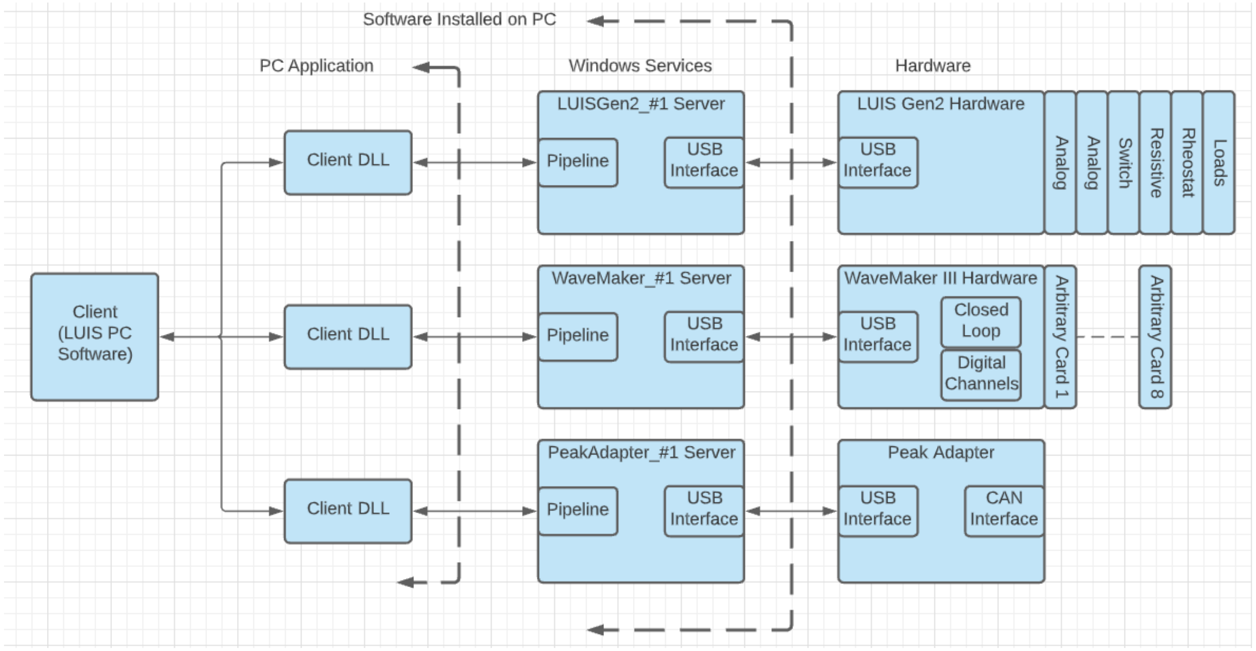


Figure 1: LUIS Gen2 Server Connections

The server connections for LUIS Gen3 hardware are similar except for the connection to the WaveMaker hardware. A LUISClient DLL is not needed to connect to the WaveMaker since all communications now are routed through the Gen3 Main Module Hardware. To accommodate the simplified connection, a new Gen3 server (LUISGen3_#1) was created. The Peak Adapter is optional and only needed if the user wants to scan J1939 protocol messages off of the CAN bus and display them on the LUIS GUI. Transmitting J1939 messages is handled by the MainModule and requires separate setup messages.

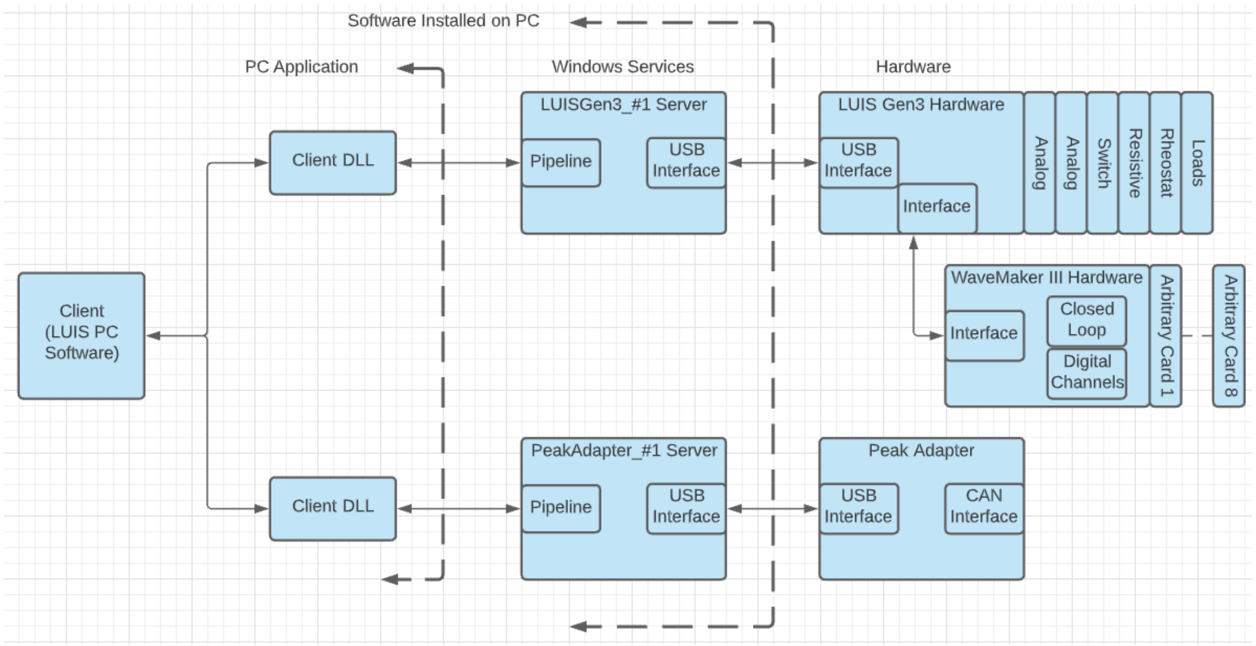


Figure 2: LUIS Gen3 Server Connections

The LUIS PC Software and Servers use Plugins to encode and decode messages. Having up to date and the correct Plugins ensures proper communications. Each piece of hardware (MainModule, Resistive, etc.,) have their own specific Plugin that provides the message encoding and decoding function.

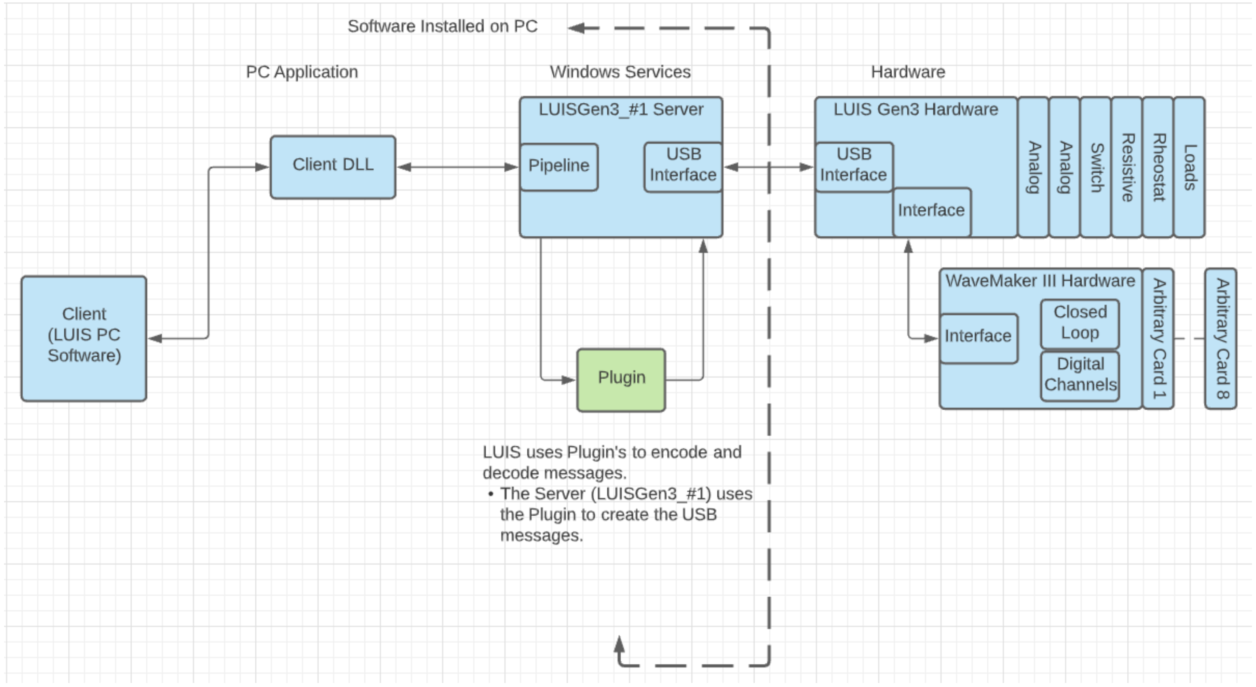


Figure 3: LUIS Plugin Usage

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.

A C# application is available to examine how the commands are used. Please request the LUIS Demo App from GarTech Enterprises.

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

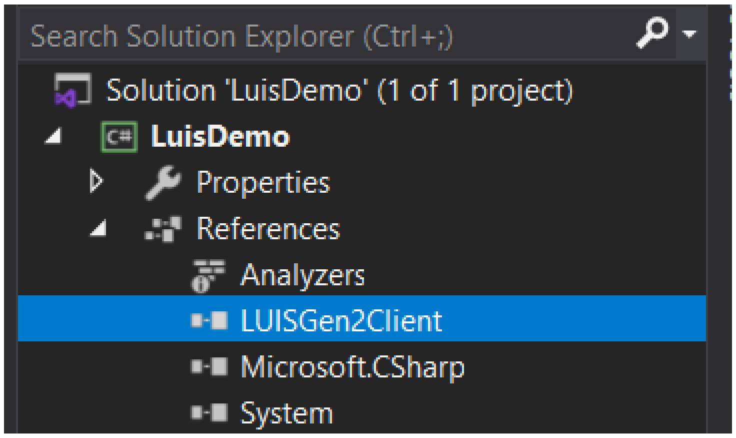


Figure 4: LUISGen2Client Reference in Visual Studio.

The DLL is used for all Client connections to the LUIS hardware. For LUISGen2 hardware, two instances must be created of the LUISClient object. One to connect to the Main system and the other to WaveMaker. For Gen3, only one LUISClient object must be created due to all the commands going to only one place instead of two.

The following is an example of connecting to one server. The LUISClient must be instantiated, events connected, and the StartService() command sent.

```
_luisGenClient = new LuisClient {ConnectionName = cbServerSelection.Text};
_luisGenClient.MessageReceived += GenClient_MessageReceived;
_luisGenClient.Disconnected    += GenClient_Disconnected;
_luisGenClient.ClientError     += GenClient_ClientError;
_luisGenClient.StartService();
```

Figure 5: Creating and Connecting to the LUISClient object Example

Once the connection has been established to the server, the client can now send commands into and receive data from the LUIS hardware. The commands are formatted String commands which are sent into the Plugin to get encoded to USB commands:

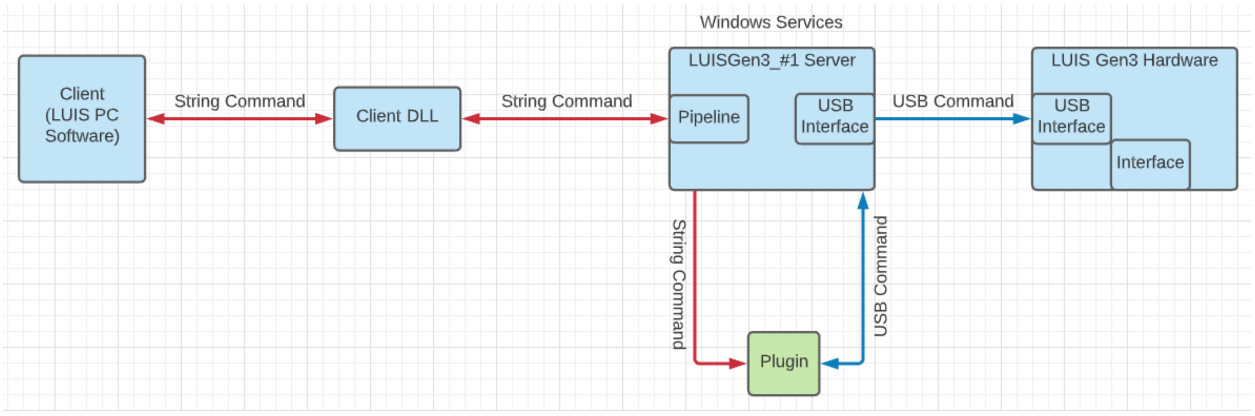


Figure 6: Command Path and Conversion

The Servers can connect to multiple clients concurrently. When a client sends a message to the Server it will be repeated to all clients that are connected. This ensures all clients are kept up to date with the hardware which can be controlled by any client.

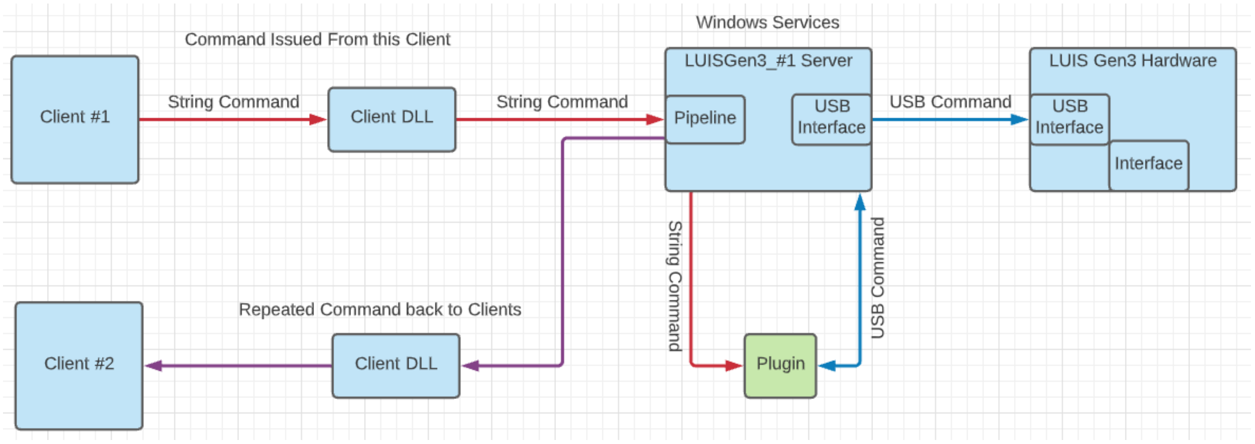


Figure 7: Reflected Commands to Clients

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:

1. A C# Example:

```
_luisGenClient.StopService();
```

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

"LUISGen3_#1":	Commands Main, Wavemaker, Analogs, Switch, Loads, Rheostat
"LUISGen2_#1":	Commands Main, Analogs, Switch, Loads, Rheostat
"WaveMaker_#1":	Commands WaveMaker, (LUISGen2 Only)
"PeakAdapter_#1":	Commands Gen1 or any CAN/J1939 message

Sending Commands

There are two different methods to send a command to the hardware:

1. `SendMsg()`
   a. Typical command used for sending messages. The commands are placed into a buffer.
   b. C# Example:
   ```
   _luisGenClient.SendMsg("Data: Main Module, VBATT Relay#1,1");
   ```
2. `SendNonBufferedMsg_single()`
   a. A message that is sent without buffering.
   b. C# Example:
   ```
   _luisGenClient.SendNonBufferedMsg_single("Data: Main Module, VBATT Relay#1,1");
   ```

The above message will turn On Relay#1 in the Main Module.

Several commands can be sent concatenated together instead of just one at a time. The pipe "|" character seperates the commands. For example, the following will send two relay commands together:

```
_luisGenClient.SendMsg("Data: Main Module, VBATT Relay#1,1|
                        Data: Main Module, VBATT Relay#2,1");
```

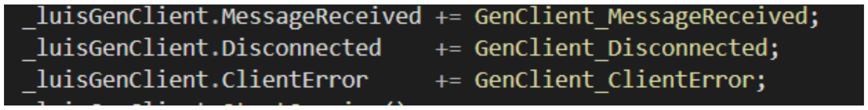
The servers will separate and buffer the commands and issue them one at a time across the USB bus.

Client Events

There are three events that can be connected to from the Client DLL. The Events are:

1. MessageReceived
   a. A message from the LUIS Hardware, this typically has been requested.
   b. A message from the Server that has been issued by another client.
2. Disconnected
   a. If the LUIS Hardware disconnects from the Server, this event is fired.
3. ClientError
   a. When an error occurs in the Server, this event is fired.

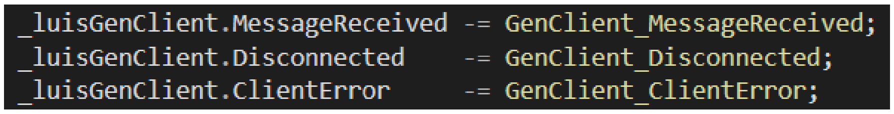
To connect to the Events, connect a function call to the event handler. The following is a C# example of event connection.



```
_luisGenClient.MessageReceived += GenClient_MessageReceived;
_luisGenClient.Disconnected    += GenClient_Disconnected;
_luisGenClient.ClientError     += GenClient_ClientError;
```

Figure 8:Connecting to Events

When the application closes, always disconnect from the events. The following is a C# example:



```
_luisGenClient.MessageReceived -= GenClient_MessageReceived;
_luisGenClient.Disconnected    -= GenClient_Disconnected;
_luisGenClient.ClientError     -= GenClient_ClientError;
```

Figure 9:Disconnect from Events

Plugin Names

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

LUIS Gen2 and Gen3 Systems

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

LUIS Gen3 Systems Only

- "Rheostat Module"
- "Selector Module"
- "MainModuleJ1939PortBPlugin"
- "MainModuleJ1939PortCPlugin"

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)”

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)”

Peak Adapter

- “PeakAdapterPlugin”

Common LUIS Gen2 and Gen3 Commands

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”

    “SwitchModule”

    “Resistive Load”

    “Rheostat Module” (LUIS Gen3 Only)

    “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”

    “SwitchModule”

    “Resistive Load”

    “Rheostat Module” (LUIS Gen3 Only)

    “WaveMaker”

Command =

“GetAllVersionData”

Requests all like modules to return version information

Example =

“GetInfo:Main Module,GetAllVersionData”

Returns the Main Modules version information

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

LUIS Gen2 and Gen3 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 Relay Number]”

Main Module Relay Numbers =

“VBATT Relay#1”

“Switched VBATT Relay#1”

. . .
"Switched VBATT Relay#4"

Example Command =

"GetData: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

Action

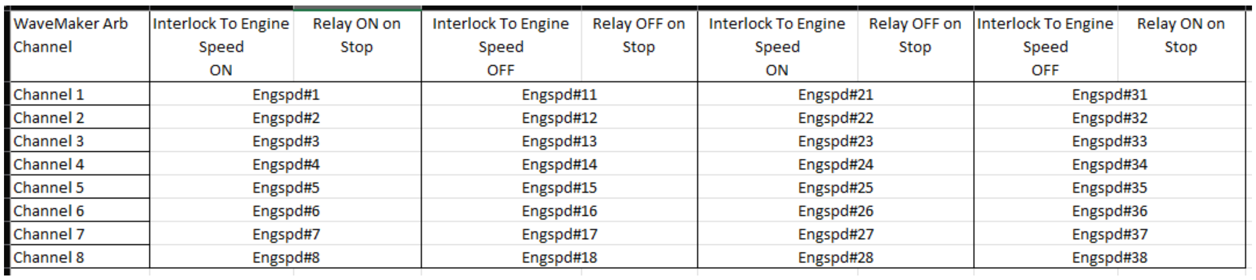
Configures the automated toggle feature so the user can toggle the mechanical relays at a desired rate. There are a few options to configure the function:

1. Decide if one of the Wavemaker channels needs to be set to a specific value
2. The end state of the Relay, this can be ON or OFF
3. If a Wavemaker channel has been selected, what Frequency should it be set for.
4. The number of cycles of ON and OFF that should be performed.
5. The ON and OFF dwell times.
6. A random dither ON and OFF time.
7. The relay to be toggled.

There are seven commands that configure the toggling feature and monitor its status.

Action Command #1 for Toggling Relay Feature (Wavemaker Settings)

The first command will setup the Wavemaker channel, interlocking of that Wavemaker channel, and the end state of the relay. The following table is the encoded value that needs to be selected.



WaveMaker Arb Channel	Interlock To Engine Speed ON	Relay ON on Stop	Interlock To Engine Speed OFF	Relay OFF on Stop	Interlock To Engine Speed ON	Relay OFF on Stop	Interlock To Engine Speed OFF	Relay ON on Stop
Channel 1	Engspd#1		Engspd#11		Engspd#21		Engspd#31	
Channel 2	Engspd#2		Engspd#12		Engspd#22		Engspd#32	
Channel 3	Engspd#3		Engspd#13		Engspd#23		Engspd#33	
Channel 4	Engspd#4		Engspd#14		Engspd#24		Engspd#34	
Channel 5	Engspd#5		Engspd#15		Engspd#25		Engspd#35	
Channel 6	Engspd#6		Engspd#16		Engspd#26		Engspd#36	
Channel 7	Engspd#7		Engspd#17		Engspd#27		Engspd#37	
Channel 8	Engspd#8		Engspd#18		Engspd#28		Engspd#38	

Figure 10: Toggle Switch Encodings

For example, if the user wants Arbitrary Channel #1 on the Wavemaker to be SET to a value AND the toggling relay to be ON at stop then the value of Engspd#1 would be selected.

Format = "Action: Main Module,6,[Encoding from Figure 10], [Set Frequency]

Encoding from Figure 10 = see Figure 10

Set Frequency = 0 to X (typically in RPM)

Example Command =

"Action: Main Module,6,EngSpd#1,500

Sets the Wavemaker CH1 to 500 RPM when toggling starts and leaves the Relay ON once completed.

Action Command #2 for Toggling Relay Feature (Dwell Time)

The second command to be sent controls the ON and OFF dwell time of the relay.

Format = "Action: Main Module,1,[Main Module Relay Number],[OFF Dwell Time],[ON Dwell Time]"

Main Module Relay Numbers =

"Relay#1"

. . .

"Relay#4"

ON and OFF Dwell Time = Values are in increments of 10ms. A value of 2 is equivalent to 20ms.

Example Command =

"Action: Main Module,1,Relay#1,1,1

Sets the Main Module Relay#1 for on ON and OFF dwell time of 10ms

Action Command #3 for Toggling Relay Feature (Dither Dwell Times)

The Third command to be sent controls the ON and OFF dither dwell time of the relay. This value will randomly add or subtract to the set dwell time for ON or OFF times.

Format = "Action: Main Module,2,[Main Module Relay Number],[OFF Dither Time],[ON Dither Time]"

Main Module Relay Numbers =

"Relay#1"

. . .

"Relay#4"

ON and OFF Dither Time = Values are in increments of 10ms. A value of 2 is equivalent to 20ms.

Example Command =

"Action: Main Module,2,Relay#1,1,1

Sets the Main Module Relay#1 for on ON and OFF dither time of 10ms

Action Command #4 for Toggling Relay Feature (Toggle Cycles)

The fourth command to be sent controls the number of toggling cycles to be performed.

Format = “Action: Main Module,3,[Main Module Relay Number],[# of Cycles]”

  Main Module Relay Numbers =

    “Relay#1”

    . . .

    “Relay#4”

  # of Cycles = the ON and OFF combined total. To excute 10 ON and 10 OFF cycles a value of 20 needs to be set.

Example Command =

  “Action: Main Module,3,Relay#1,20

  Sets the Main Module Relay#1 for 10 cycles ON and 10 cycles OFF

The following is an example of sending all four commands to configure the feature and start executing:

```
Action:Main Module,6,Engspd#1,500
Action:Main Module,1,Relay#1,2,1
Action:Main Module,2,Relay#1,1,2
Action:Main Module,3,Relay#1,20
```

Action Command #5 for Toggling Relay Feature (Count Request)

The fifth command will request the current toggle count that the Main Module is on. This request can be used to know how much longer the feature will run.

Format = “Action: Main Module,7,CurCounts”

Example Command =

  “Action: Main Module,7,CurCounts

  Requests the current count of the toggling feature

The Main Module will respond to the request with the amount of cycles that are left to do.

Format = “Action: Main Module,[Relay#],[Remaining Counts]”

Example Command Response =

  “Action: Main Module,1,115”

  There are 115 cycles left to do for Relay#1

Action Command #6 for Toggling Relay Feature (Stop)

The sixth command will request the toggling feature to Stop.

Format = “Action: Main Module,9,STOP”

Example Command =

“Action: Main Module,9,STOP”

Requests the toggling feature to stop running

Action Command #7 for Toggling Relay Feature (Pause)

The seventh command will request the toggling feature to Pause.

Format = “Action: Main Module,8,PAUSE”

Example Command =

“Action: Main Module,8,PAUSE”

Requests the toggling feature to pause.

Analog Output Module

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 Output,Analog Output#1”

Gets the value of the first Analog channel in counts.

Switch Module

Data

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

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

  Switch Numbers =

    “Switch#1”

    . . .

    “Switch#161”

  Value = 0 or 1

Example Command:

  “Data: SwitchModule, Switch#1,1”

  Turns ON the first Switch of the Switch modules

GetData

Requests the current status of the hardware value.

Format = ”GetData:SwitchModule,[Switch Number]”

  Switch Numbers =

    “Switch#1”

    . . .

    “Switch#161”

Example Command =

  “GetData:SwitchModule, Switch#1”

  Get the value of Switch #1

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”

  “GetDataBroadcast:Resistive Load,Resistive Load#36”

DataMux (Response from GetDataBroadcast Command)

The response from hardware that contains the status of the loads. The bytes are encoded to the load number.

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

"36" = 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.

Rheostat Module (LUIS Gen3 Only)

Data

Sets the hardware value to the value in the command. The destination name comes from the Plugin.
Format = "Data:Rheostat Module,[Rheostat Number],[Value]"

Rheostat Numbers =

"Rheostat #1"

. . . .

"Rheostat #64"

Value = hardware value (0- 4294967295)

Example Command =

"Data:Rheostat Module,Rheostat #1,1000"

Sets the Rheostat #1 value to 1000.

GetData

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

Rheostat Numbers =

"Rheostat #1"

. . . .

"Rheostat #64"

Example Command =

“GetData: Rheostat Module,Rheostat #1”
Gets the value for the Rheostat #1 channel.

Selector Module (LUIS Gen3 Only)

Data

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

Selector Numbers =
“Selector #1”
Value = hardware value (1-4)

Example Command =
“Data:Selector Module,Selector #1,2”
Sets the Selector #1 value to 2.

GetData

Requests the current status of the hardware value.
Format = “GetData: Selector Module,[Selector Number]
Selector Numbers =
“Selector #1”

Example Command =
“GetData: Selector Module, Selector #1”
Gets the value for the Selector #1 channel.

MainModuleJ1939PortAPlugin
MainModuleJ1939PortBPlugin (LUIS Gen3 Only)
MainModuleJ1939PortCPlugin (LUIS Gen3 Only)

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

0 = Main Module Port A

1 = Main Module Port B

2 = Main Module Port C

Example Command =

“CANConfig:MainModuleJ1939PortCPlugin,418316794,20,8,2

Transmit 0x18EF01FA message every 20mS on Main Port C CAN bus

ClearList

Clears the list of CAN messages on the Main Port C bus.

Format = “ClearList:MainModuleJ1939PortAPlugin,0”

Format = “ClearList:MainModuleJ1939PortBPlugin,0”

Format = “ClearList:MainModuleJ1939PortCPlugin,0”

StartBroadcast

Starts the CAN message table broadcasting if messages are created.

Format = “StartBroadcast:MainModuleJ1939PortAPlugin,0”

Format = “StartBroadcast:MainModuleJ1939PortBPlugin,0”

Format = “StartBroadcast:MainModuleJ1939PortCPlugin,0”

StopBroadcast

Starts the CAN message table broadcasting if messages are created.

Format = “StopBroadcast:MainModuleJ1939PortAPlugin,0”

Format = “StopBroadcast:MainModuleJ1939PortBPlugin,0”

Format = “StopBroadcast:MainModuleJ1939PortCPlugin,0”

SetBusSpeed (LUIS Gen2 Only)

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.

GetBusSpeed (LUIS Gen3 Only)

Gets the status of the CAN bus.

Format = “GetBusSpeed:MainModuleJ1939PortAPlugin,0”

Format = “GetBusSpeed:MainModuleJ1939PortBPlugin,0”

Format = “GetBusSpeed:MainModuleJ1939PortCPlugin,0”

GetBusSpeed Response(LUIS Gen3 Only)

When the GetBusSpeed command is executed, the Main module will respond with the status of that CAN bus.

Format = “CanBusStatus:[Port],[Bus Speed],[Status]

Port =

“MainModuleJ1939PortAPlugin”

“MainModuleJ1939PortBPlugin”

“MainModuleJ1939PortCPlugin”

Bus Speed =

“250000”

“500000”

Status =

“Connected”

“Listen Only”

“BUS Error”

Example Response Command =

“CanBusStatus:MainModuleJ1939PortCPlugin,500000,Connected”

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 = ”GetData:WaveMaker,[Channel]”
Channel =
“Arbitrary_CH#1”
. . .
“Arbitrary_CH#8”
“Digital_CH#1”
. . .
“Digital_CH#10”
Example Command =
“GetData: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

Example Commands To Configure All Channels =

```
ChannelConfig:WaveMaker,Arbitrary_CH#1,0,0,0,Hall,Rpm,2,Master,0,
ChannelConfig:WaveMaker,Arbitrary_CH#2,0,0,0,Hall,Rpm,2,Slave,0,Arbitrary_CH#1
ChannelConfig:WaveMaker,Arbitrary_CH#3,0,0,0,Hall,Rpm,2,Master,0,
ChannelConfig:WaveMaker,Arbitrary_CH#4,0,0,0,Hall,Rpm,2,Master,0,
ChannelConfig:WaveMaker,Arbitrary_CH#5,0,0,0,Hall,Rpm,2,Master,0,
ChannelConfig:WaveMaker,Arbitrary_CH#6,0,0,0,Hall,Rpm,2,Master,0,
```

ChannelConfig:WaveMaker,Arbitrary_CH#7,0,0,0,Hall,Rpm,2,Master,0,
ChannelConfig:WaveMaker,Arbitrary_CH#8,0,0,0,Hall,Rpm,2,Master,0,
ChannelConfig:WaveMaker,Digital_CH#1,1,1,0,Hall,Freq,0,Master,0,
ChannelConfig:WaveMaker,Digital_CH#2,0,1,0,Hall,Freq,0,Master,0,
ChannelConfig:WaveMaker,Digital_CH#3,0,1,0,Hall,Freq,0,Master,0,
ChannelConfig:WaveMaker,Digital_CH#4,0,1,0,Hall,Freq,0,Master,0,
ChannelConfig:WaveMaker,Digital_CH#5,0,1,0,Vr,Freq,0,Master,0,
ChannelConfig:WaveMaker,Digital_CH#6,0,11,0,Hall,Rpm,0,Master,0,
ChannelConfig:WaveMaker,Digital_CH#7,0,1,0,Hall,Freq,0,Master,0,
ChannelConfig:WaveMaker,Digital_CH#8,0,1,0,Hall,Freq,0,Master,0,
ChannelConfig:WaveMaker,Digital_CH#9,0,1,0,Hall,Freq,0,Master,0,
ChannelConfig:WaveMaker,Digital_CH#10,0,1,0,Hall,Freq,0,Master,0,

Note: The channel commands can all be sent together but a "|" character must separate them.

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.

Note: Long waveforms take additional processing time to load. To determine when the channel is loaded and WaveMaker is ready for another ChannelDataLoad command, send the following message:

```
Server:ResponseRequest,QueEmpty
```

This message should be concatenated to the end of the waveform data with the "|" character as the separator. Once the channel has been loaded, a QueEmpty response will be sent from WaveMaker back to the clients via the Message Receive Event.

ClosedLoopData

There are several commands that control and setup the Closed Loop feature for LUIS. In Gen3, these commands are all directed to the Gen3 server whereas in Gen2 these commands are sent to the WaveMaker server.

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],[Option1],[Option2]"

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)

Example:

```
ClosedLoopData:WaveMaker,Arbitrary_CH#1,Gain,100
```

Set the Gain for 100

"**PercentLoad**" = Adjust the load on the engine (-125% to +125%)

Example:

```
ClosedLoopData:WaveMaker,Arbitrary_CH#1,PercentLoad,0
```

Set the Percent load to 0

"**ClosedLoop**" = Run the system in a Closed loop mode

Option1 = ECM CAN Source ID, Typically 0

Option2 = Application Type (0=OnRoad, 1=Industrial)

Note: OnRoad is LUIS Gen3 compatible only.

Example:

```
ClosedLoopData:WaveMaker,Arbitrary_CH#1,ClosedLoop,0,0
Turn ON closed loop, ECM ID=0, OnRoad Application model
```

"**OpenLoop**" = Run the system in an Open loop mode

Example:

```
ClosedLoopData:WaveMaker,Arbitrary_CH#1,OpenLoop,0,0
Turn OFF closed loop, ECM ID=0, OnRoad Application model
```

Note: OnRoad is LUIS Gen3 compatible only.

"**Start**" = Start the engine

Example:

```
        ClosedLoopData:WaveMaker,Arbitrary_CH#1,Start
        Start the engine model
```

"**Reset**" = Reset the model

Example:

```
        ClosedLoopData:WaveMaker,Arbitrary_CH#1,ESS_BCST,1
        Resets the model
```

The following Closed Loop commands are LUIS Gen3 compatible only and the Application Model must be OnRoad.

"**VehicleWeight**" = In (k) US Pounds, 20 is 20000 lbs

Example:

```
        ClosedLoopData:WaveMaker,Arbitrary_CH#1,VehicleWeight,44
```

Set the vehicle weight to 44k lbs.

"**Grade**" = The grade the vehicle is on (-15% to 15%)

Example:

```
        ClosedLoopData:WaveMaker,Arbitrary_CH#1,Grade,-5
```

Set the grade to -5%

"**VssSetup**" = Setup the VSS output

Option1 = Wavemaker output channel for VSS

Option2 = Current Transmission Ratio

Example:

ClosedLoopData:WaveMaker,Arbitrary_CH#1,VssSetup,Digital_CH#1,100,1

Set the Wavemaker VSS output to Digital Channel #1, current transmission ratio is 100 (100=Neutral), if transmission ratio is not 100 then that indicates the transmission is in gear. A ratio of 3.10 would be sent as a value of 310.

ClosedLoopData:WaveMaker,Arbitrary_CH#1,VssSetup,Digital_CH#1,310,1

"**VssSensorType**" = the sensor settings for the VSS channel in the Wavemaker.

Note: These settings will override the Wavemaker configuration.

Example VR:

ClosedLoopData:WaveMaker,Arbitrary_CH#1,VssSensorType,Digital_CH#1,VR

The Vss sensor is a VR type sensor. Options include VR, HALL, J1939.

Example of J1939 SPN 191 Message:

ClosedLoopData:WaveMaker,Arbitrary_CH#1,VssSensorType,J1939#20,J1939

Example of J1939 SPN 1623 Message:

ClosedLoopData:WaveMaker,Arbitrary_CH#1,VssSensorType,TACH#21,TACH

"**VssRatios**" = The rear axle ratio and the tire size.

Option1 = Rear Axle * 100

Option2 = Tire size

Example:

```
ClosedLoopData:WaveMaker,Arbitrary_CH#1,VssRatios,Digital_CH#5,355,311
```

"**VssRequest**" = Sends in the current Transmission Ratio and elicits a response from the LUIS hardware.

Example:

A typical response from Wavemaker will be:

ClosedLoopMsg:VssRequestResponse,TailshaftSpeed,0,VehicleSpeed,0,EngineSpeed,0

An example of a typical setup of the Closed Loop model for LUIS Gen3:

```
ClosedLoopData:WaveMaker,Arbitrary_CH#1,ClosedLoop,0,0
ClosedLoopData:WaveMaker,Arbitrary_CH#1,Gain,100
ClosedLoopData:WaveMaker,Arbitrary_CH#1,PercentLoad,0
ClosedLoopData:WaveMaker,Arbitrary_CH#1,VehicleWeight,44
ClosedLoopData:WaveMaker,Arbitrary_CH#1,Grade,0
ClosedLoopData:WaveMaker,Arbitrary_CH#1,VssRatios,Digital_CH#5,355,311
ClosedLoopData:WaveMaker,Arbitrary_CH#1,VssSensorType,Digital_CH#5,VR
ClosedLoopData:WaveMaker,Arbitrary_CH#1,VssSetup,Digital_CH#5,100,1
```

Note: The messages can all be sent together but must contain a pipe "|" between each message. Once the setup messages have been sent, the client can then "Start" the engine model and start polling the "VssRequest" with new transmission ratios and receiving the responses back from the engine model.

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%

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