QML Viewer Plugin Reference Manual

G2C V1

Reach Technology

09/21/2017

Reach Technology

www.reachtech.com
www.linuxtouchscreen.com

Sales: 408-754-4176
sales@reachtech.com

Technical Support: 503-675-6464
techsupport@reachtech.com

1 Overview

This document describes the QML Viewer Plugins available for the Reach Technology G2C Display Modules. There are built-in plugins in the QML Viewer, and external plugins that are loaded when needed.

2 Built-in Plugins

2.1 Connection.sendMessage

Use this javascript code to send a message to a MCU through the module serial port (RS232 or RS485), CAN port or TCP/IP port (depending on agent running):

```
connection.sendMessage(“w=15”);
```

2.2 Connection.enableHeartbeat

To enable a heartbeat response from the module use:

```
 // Send the string “ping” every 5 seconds to the MCU through the
  // module serial port (RS232 or RS485), CAN port or TCP/IP port.
  connection.enableHeartbeat(5);

  // Send the string “ping” every 5 seconds to the MCU through the
  // module serial port (RS232 or RS485), CAN port or TCP/IP port.
  // When “pong” is received the interval will restart to every 5
seconds.
  connection.enableHeartbeat(5, “ping”, “pong”);
```

2.3 Connection.disableHeartbeat

To disable a heartbeat response from the module use:

```
connection.disableHeartbeat(); // Disable the heartbeat.
```

2.4 Connection.enableLookupAck

To enable a QML object.property lookup response from the module use:

```
connection.enableLookupAck();
```

The responses are as follows:

A. LUNO: Lookup no object found.
B. LUMP: Lookup no property found.
C. LUOK: Lookup ok.
D. SYNERR: Message syntax error.

2.5 Connection.disableLookupAck

To disable a lookup response from the module use:

```
connection.disableLookupAck();  // Disable the lookup response.
```

2.6 Settings.setValue

To set a key/value pair in the application.conf file to persist data us:

```
settings.setValue("volume", 100);  // Sets the key volume to 100.
```

2.7 Settings.getValue

To retrieve a value for a certain key in the applications.conf file use:

```
var volume = settings.getValue("volume");  // Get the value for key volume.
```

2.8 Settings.remove

To remove a key from the application.conf file use:

```
settings.remove("volume");  // Remove the key volume.
```

2.9 Screen.save

To save a screen shot to the file system use (png and jpg image formats are supported):

```
screen.save("/application/images/screen.png");
```

3 System Plugin

The System is the only external plugin currently defined.

File: libsystemplugin.so

Add this line to your qml file to use the System plugin:

```
Import ”components”
```

3.1 Backlight QML Element

Qml usage:

```
Backlight {
   id: backlight
 }
```

Javascript usage:

```
var brightness = backlight.brightness();  // Returns the current brightness of the screen in the range of 0-100.

backlight.disable();  // Disables the backlight and darkens the screen

backlight.enable();  // Enables the backlight

backlight.setBrightness(50);  // Sets the brightness of the screen.  Use values 0 - 100.
```

3.2 Beeper QML Element

Qml usage:

```
Beeper {
   Id: beeper
}
```

Javascript usage:

```
beeper.setVolume(50);   // Sets the volume of the beeper.  Use values 0 - 100.

beeper.setDuration(100);  // Sets the duration of the beeper in ms.

beeper.setFrequency(300);  // Sets the frequency of the beeper.

beeper.beep();

var volume = beeper.volume();  // Get the duration of the beeper.
var duration = beeper.duration();  // Get the duration of the beeper.

var frequency = beeper.frequency();  // Get the frequency of the beeper.
```

3.3 GPIOPinInput QML Element

QML usages:

```
GPIOPinInput {
   id: pin1
   pin: 1              // pin to read
   pollPin: false    //  poll the pin and emit a signal called stateChanged
   state: 0            //  set the initial state of the pin 0 off, 1 on
}

GPIOPinInput {
   id: pin1
   pin: 1              // pin to read
   pollPin: true      //  poll the pin and emit a signal called stateChanged
   state: 0            //  set the initial state of the pin 0 off, 1 on

   onStateChanged :{
      if (state == 1)
         runSomeFunction();
   }
}
```

Javascript usage:

```
pin1.readPin();  // Read pin and sets state.
var statePin1 = pin1.state();  // Get state.
```

3.4 GPIOPinOutput QML Element

QML usage:

```
GPIOPinOutput {
   id: pin1
   pin: 1              // pin to write and read
}
```

Javascript usage:

```
pin1.writeToPin(1);  // write 1 to pin 1
var test = pin1.readPin();  // read from pin
```

3.6 GPIOPinsInput QML Element

QML usage:

```
GPIOPinsInput {
   id: pins

   onStateChanged: {
      console.debug(state);
   }
}
```

Javascript usage:

```
pins.readRegister();  // read register
console.debug(pins.state);
```

3.73 GPIOPinsOutput QML Element

QML usage:

```
GPIOPinsOutput {
   id: pins
}
```

Javascript usage:

```
var value= pins.readPins();  // read pins
console.debug(value);
```

3.8 Network QML Element

QML usage:

```
Network {
   id: network
}
```

Javascript usage:

```
var ip = network.getIp();
console.debug(ip);
```

3.10 SqLite QML Element

QML usage:

```
SqLite {
   id: db
}
```

Javascript usage:

```
If (db.openDB()) {  // open sqlite database

   // create a table to store persistent data
   db.execSql("CREATE TABLE IF NOT EXISTS Person(ID INTEGER PRIMARY KEY
AUTOINCREMENT, NAME TEXT NOT NULL, AGE INT NOT NULL, ADDRESS CHAR(100)");

   // insert data into the Person table
   db.execSql("insert into Person (NAME, AGE, ADDRESS) values (\"John Doe\", 21, \"100 Main St
Portland OR\");"

   // query Person table
   var rows = db.getRows("select NAME, ADDRESS from Person");

   for (var i=0; i < rows.length; i++) {
      var data = rows[i];
      // output NAME and ADDRESS to the console
      console.debug(data[0] + " " + data[1]);
   }

   db.closeDB();  // close sqlite database

}
else {
   console.debug(db.lastError());  // output sqlite database error if we cannot open the db
}
```

3.11 System QML Element

QML usage:

```
System {
    id: system
}
```

Javascript usage:

```
var date = system.currentDataTime(); // get current date time

 system.setDate(2018, 1,10);  // set the system date to January 10th 2018

 system.setTime(12, 10, 30);  // set the system time to 12:10:30 pm

 // execute fb2image application with parameters
 system.execute(“/application/bin/fb2image”, “-f”, “/application/screenshot.jpg”);

 // reboot the module
 system.execute("reboot");

  // execute a shell command
  var ret = system.shell(“cd /application/bin && ls –l”);
  console.debug(ret);
```

3.12 Upgrade QML Element

Upgrade element can be used to upgrade an existing qml application from another file source like a USB stick.

QML usage:

```
Upgrade {
    id: upgradeObject
    upgradeSourcePath: “/run/media/sda1/application/src”   // usb stick
    applicationSourcePath: “/application/src”  // module qml application path

    onProgressChanged : {
        progressBar.value = progress();  // updates a progress bar to show user that an upgrade is in
progress
    }

    onErrorMessageChanged: {
        message.text = errorMsg();  // updates a text label to show the user an error has occurred
    }
}


 upgradeObject.upgrade();  // starts the upgrade process of deleting and copying files
```