

|   |                                                                                                                                                                                         |                     |              |            |
|---|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------|--------------|------------|
| ↑ | <b>Software Release Note for Sensor Management<br/>Computer (SMC) Core Components version 2.5.1</b>  |                     |              |            |
|   | responsible engineer                                                                                                                                                                    | A. Trapani          | created      | 02/01/2016 |
|   | product name                                                                                                                                                                            | SMC Core Components | last updated | 02/01/2016 |
|   | software version                                                                                                                                                                        | 2.5.1               | Status       | Final      |
|   | software part number                                                                                                                                                                    | 02617-34            |              |            |
|   | document part number/rev                                                                                                                                                                | 02619-34 Rev A      |              |            |

## TABLE OF CONTENTS

|                                                                                                  |    |
|--------------------------------------------------------------------------------------------------|----|
| Overview .....                                                                                   | 3  |
| Changes in Version 2.5.1 (02/01/2016, P/N 02617-34) .....                                        | 5  |
| New Features .....                                                                               | 5  |
| Bug Fixes .....                                                                                  | 6  |
| SWSMAC-552 "PPPHost payload registration does not go through, cannot communicate with VMC" ..... | 6  |
| Dependencies .....                                                                               | 9  |
| Open Issues .....                                                                                | 9  |
| End-user Impacts .....                                                                           | 9  |
| Support Impacts .....                                                                            | 9  |
| Installation/Upgrade Instructions .....                                                          | 10 |
| Setting up the SD Card .....                                                                     | 12 |
| Working with the SMC and the SV2 Float C&C .....                                                 | 12 |
| Working with the SMC and the SV3 via Sensor Port.....                                            | 14 |
| Working with the SMC and the SV3 via Expansion Port .....                                        | 15 |
| Installing the Application .....                                                                 | 18 |
| Changing Encryption Passphrases.....                                                             | 20 |
| Component Configuration .....                                                                    | 21 |
| Aquatracka (Chelsea Technologies Group) Configuration.....                                       | 23 |
| Commands .....                                                                                   | 23 |
| BenthosModem Configuration (not supported).....                                                  | 25 |
| Configuring the Client Program.....                                                              | 25 |
| Loading the Modem Firmware .....                                                                 | 25 |
| Configuring Modem Parameters .....                                                               | 26 |
| C3 (Turner Designs) Configuration.....                                                           | 27 |
| Commands .....                                                                                   | 27 |
| Events and Telemetry .....                                                                       | 27 |
| Common XML file keys to edit .....                                                               | 29 |

|                                                               |    |
|---------------------------------------------------------------|----|
| ConnectionManager Configuration .....                         | 29 |
| Common XML file keys to edit .....                            | 29 |
| DataRecorder Configuration .....                              | 30 |
| Common XML file keys to edit .....                            | 30 |
| Using the DataRecorder.....                                   | 31 |
| GPSListener Configuration .....                               | 32 |
| Commands .....                                                | 32 |
| CORBA Events.....                                             | 33 |
| XML Configuration .....                                       | 34 |
| Logger.....                                                   | 36 |
| GPSWaves Configuration (GPS and IMU based waves sensor )..... | 36 |
| Commands .....                                                | 37 |
| CORBA Events.....                                             | 37 |
| XML Configuration .....                                       | 38 |
| LuaWrapper Configuration .....                                | 39 |
| Commands .....                                                | 39 |
| Special Variables .....                                       | 40 |
| CORBA Event Tables and Variable Definitions .....             | 43 |
| Pre-defined Callback Functions .....                          | 45 |
| XML Configuration .....                                       | 46 |
| Sample lua script.....                                        | 46 |
| Magnetometer Configuration (Marine Magnetics Explorer).....   | 48 |
| Commands .....                                                | 48 |
| CORBA Events.....                                             | 50 |
| XML Configuration .....                                       | 51 |
| MAM Configuration (Datawell MOSE).....                        | 51 |
| Commands .....                                                | 51 |
| Common XML file keys to edit .....                            | 53 |
| PayloadInterface Configuration.....                           | 53 |
| PayloadInterface Commands.....                                | 54 |
| "hostid.txt" File .....                                       | 58 |
| PPPHost Configuration .....                                   | 58 |
| Common XML file keys to edit .....                            | 58 |
| One-Time Call-in Execution .....                              | 59 |
| Configuring the RUDICS Modem .....                            | 59 |
| Configure the PPP connection script .....                     | 60 |
| Configure and verify RUDICS call parameters .....             | 60 |
| PPPHost Commands.....                                         | 64 |

|                                                          |     |
|----------------------------------------------------------|-----|
| Manifest File Metadata .....                             | 64  |
| Route Configurations .....                               | 65  |
| Tips for Script Developers.....                          | 65  |
| SeabirdGPCTD Configuration.....                          | 66  |
| SeabirdGPCTD Telemetry Output .....                      | 68  |
| SeabirdGPCTD Commands .....                              | 70  |
| SimpleSBDQueue Configuration.....                        | 71  |
| Commands .....                                           | 71  |
| CORBA Events.....                                        | 72  |
| XML Configuration .....                                  | 73  |
| TempPress Configuration.....                             | 74  |
| Server Run Modes .....                                   | 74  |
| Commands .....                                           | 75  |
| Alarms .....                                             | 82  |
| CORBA Events.....                                        | 83  |
| Telemetry Sub-Type 0 (deprecated).....                   | 84  |
| Telemetry Sub-Type 1.....                                | 85  |
| Other Things to Note.....                                | 88  |
| timelimit2 .....                                         | 89  |
| VemcoVR2C Configuration .....                            | 89  |
| Selected XML file keys.....                              | 89  |
| Using the VemcoVR2C.....                                 | 90  |
| VirtualRelay Configuration .....                         | 91  |
| Common XML file keys to edit .....                       | 91  |
| Typical Virtual Relay configurations .....               | 93  |
| Commands .....                                           | 93  |
| WetLabsSensorSimulator.....                              | 94  |
| Appendix 1 – Sample Application Configuration File ..... | 95  |
| Appendix 2 – Sample Telemetry Configuration .....        | 98  |
| History .....                                            | 113 |

## Overview

The Sensor Management Computer (SMC) core components encompass the superset of standard applications, libraries, scripts and configuration files that might be loaded onto the embedded SMC Linux computer. These include:

| COMPONENT  | DESCRIPTION                                 | VERSION |
|------------|---------------------------------------------|---------|
| Aquatracka | Interface to Chelsea Aquatracka fluorimeter | 1.1.8+  |

| COMPONENT           | DESCRIPTION                                                                                                                                                       | VERSION |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|
| BackupMissionData   | A script that streams a backup of the XDATA partition and typical directories under /home/smc; used in preparation for upgrade or purging data.                   | 2.5.0+  |
| BenthosModem        | Interface to Benthos acoustic modems (tested with version 8.2.2) (not supported)                                                                                  | 1.0.10  |
| busy                | A utility that can be invoked from script to keep the CPU from going idle (busy on) or to turn that mode off (busy off)                                           | 1.1.8+  |
| C3                  | Interface to Turner Designs C3 fluorometer using command mode                                                                                                     | 1.3.0+  |
| catgps              | This is a utility that can be used during board setup or troubleshooting to configure and show the output of the GPS module when the SMC software is not running. | 1.1.8+  |
| ConnectionManager   | Listens to requests for and manages TCP/IP connections, such as a cell modem                                                                                      | 1.1.9+  |
| ConfigOctalUartLine | Configures an octal UART port electrical protocol (must be executed as <b>root</b> )                                                                              | 1.0.9+  |
| ConfigurePower      | Configures the power-saving features of the platform (must be executed as <b>root</b> )                                                                           | 1.0.9+  |
| ConfigureRUDICSCall | Configures RUDICS account, server, phone number and credentials and writes ".rudics" file                                                                         | 2.4.0+  |
| DataRecorder        | A module capable of archiving received serial port data and making it available for extraction at runtime                                                         | 1.1.0+  |
| DemoSensor          | An example sensor implementation                                                                                                                                  | 1.0.9+  |
| DisableAutoEthernet | A script that modifies the startup configuration to <b>not</b> enable eth0 automatically                                                                          | 2.1.0+  |
| EnableAutoEthernet  | A script that modifies the startup configuration to enable eth0 automatically                                                                                     | 2.1.0+  |
| EnableEthernet      | Activates the Ethernet interface and establishes an IP address via DHCP                                                                                           | 1.0.9+  |
| EventManager        | CORBA-based event subscription and distribution service                                                                                                           | 1.0.9+  |
| GetHostID           | Determines the host ID field to be used with the REST interface                                                                                                   | 2.5.0+  |
| GPSListener         | Interface to GPS receiver                                                                                                                                         | 1.0.9+  |
| GetVMCAddress       | Determine the address to be used to contact the VMC for the REST interface                                                                                        | 2.1.0+  |
| GPSWaves            | Wave height sensor based on GPS or IMU input                                                                                                                      | 2.0.1+  |
| InitOctalUartLines  | Configures the electrical protocol for all octal UART ports (must be executed as <b>root</b> )                                                                    | 1.0.9+  |
| InitPowerSwitches   | Configures the power switches to their initial (off) state (must be executed as <b>root</b> )                                                                     | 1.0.9+  |
| killsmc             | Stops all SMC processes (must be executed as <b>root</b> )                                                                                                        | 1.0.9+  |
| LISST               | Interface to Sequoia Instruments' LISST sensor                                                                                                                    | 1.7.0+  |
| Logger              | A utility program that accepts text from standard input and writes to a log file set with configurable maximum size                                               | 1.1.0+  |
| LogListener         | Listener for reported log events from the Float C&C                                                                                                               | 1.1.2+  |
| LuaWrapper          | Wrapper for a lua interpreter instance: allows execution of lua scripts                                                                                           | 1.1.2+  |
| Magnetometer        | Interface to Marine Magnetics' Explorer Magnetometer                                                                                                              | 1.7.0+  |
| MakeThumbnail       | Creates a reduced resolution image of a picture.                                                                                                                  | 1.1.8+  |
| MakeZoom            | Creates a cropped image from a picture.                                                                                                                           | 1.1.8+  |
| MAM                 | Interface to Datawell motion sensor                                                                                                                               | 1.0.9+  |
| MobileCtrl          | Mobile WGMS controller, which enables local piloting through a radio network ( <i>alpha version, not documented</i> )                                             | 2.0.1+  |
| omniNames           | CORBA (omniORB) name server                                                                                                                                       | 1.0.9+  |
| PayloadInterface    | Implementation of Float C&C payload protocol, which exposes services registered via CORBA to the Float C&C and WGMS                                               | 1.0.9+  |
| PayloadManager      | A CORBA names registry that can be enumerated                                                                                                                     | 1.0.9+  |
| PPPHost             | Manager of a point-to-point protocol connection (typically RUDICS) and executor of configured scripts while the internet connection is open                       | 1.0.9+  |
| PurgeMissionData    | Deletes the data from the XDATA partition and the common                                                                                                          | 2.5.0+  |

| COMPONENT              | DESCRIPTION                                                                                                       | VERSION |
|------------------------|-------------------------------------------------------------------------------------------------------------------|---------|
|                        | locations under /home/smc                                                                                         |         |
| ReportBoardVersion     | Reports the version number of the SMC board ("ReportBoardVersion" or "ReportBoardVersion -v")                     | 1.1.8+  |
| SeabirdGPCTD           | Interfaces a Seabird GPCTD (Glider Payload CTD)                                                                   | 2.3.0+  |
| ServerExec             | A utility that contacts an application via its CORBA-registered name and invokes a command, returning any output. | 1.0.9+  |
| SetPowerSwitch         | Turns one of the SMC power switches on or off (must be executed as <b>root</b> )                                  | 1.0.9+  |
| SetRootConfigs         | Configures which processes will start up by adding or removing .disabled files                                    | 2.4.0+  |
| SimpleSBDQueue         | Interface to an SMC-local 9522B/9523 Iridum SBD modem                                                             | 1.7.0+  |
| TakePicture            | Powers on an attached camera, sets zoom level, snaps a picture and downloads the image.                           | 1.1.8+  |
| TempPress              | Interface to temperature and pressure sensor                                                                      | 2.5.0+  |
| timelimit2             | Utility that limits runtime to a maximum time                                                                     | 1.1.2+  |
| VemcoVR2C              | Interface to the Vemco VR2C sensor                                                                                | 1.1.3+  |
| VerifyRUDICSCall       | Exercises a RUDICS call and adds an appropriate entry to the "known_hosts" file                                   | 2.4.0+  |
| VirtualRelay           | Uses a TCP/IP connection to send and receive messages from WGMS                                                   | 1.1.9+  |
| WetLabsSensor          | Interface to Wet Labs sensors like Eco Puck and CStar                                                             | 1.0.10+ |
| WetLabsSensorSimulator | Generate simulated Eco Puck samples over a pseudo serial port.                                                    | 1.1.0+  |
| WHOIMM                 | Interface to a WHOI Micro Modem                                                                                   | 1.2.0+  |

Note that this distribution is intended for those who are comfortable with Linux and making configuration changes. Other installation kits for specific configurations are also available.

## Changes in Version 2.5.1 (02/01/2016, P/N 02617-34)

This release fixes a bug between SMC 2.5.0 with PPPHost (payload-based RUDICS modem) and Regulus 1.0.3

### New Features

No new features this release

## Bug Fixes

### SWSMAC-552 "PPPHost payload registration does not go through, cannot communicate with VMC"

#### Description

Starting in SMC version 2.5.0, there is a new section in the PPPHost payload registration JSON that describes the RUDICS modem:

```
{
  "RUDICS" : {
    "KEYNAME" : "miketestbox2",
    "PHONENUM" : "88160000617",
    "SERVERDIR" : ".",
    "SERVERLOGIN" : "miketestbox2@ggsmac"
  },
  "ui" : [
    {
      "label" : "Call Now",
      "method" : "CallNow",
      "template" : "%tCALLNOW",
      "timeout" : 600,
      "tooltip" : "Attempt to make a RUDICS connection now (and perform default actions)"
    },
    {
      "label" : "Get Key Name",
      "method" : "GetKeyName",
      "template" : "%tGETKEY",
      "timeout" : 600,
      "tooltip" : "Reads the configured key name"
    }
  ]
}
```

The SMC 2.5.0 software was tested against Regulus version 1.0.1 and 1.0.2; but starting in Regulus version 1.0.3, Regulus throws an exception during the registration process that aborts the connection attempt from the SMC to the VMC:

```
2016-01-19T00:14:45.128Z@37.4108338,-122.0043498 ? rudder: setDesiredHeading:
Rudder port not ready
2016-01-19T00:14:45.151Z@37.4108338,-122.0043498 rudder: not sending rudder
alarm: delta=120086
2016-01-19T00:14:47.190Z@37.4108375,-122.0043515 * CFHashMap:34: *****
CFHashMap overwrite!!!
2016-01-19T00:14:47.214Z@37.4108375,-122.0043515 scripting: Add
{template=%trunshell N%lp, paramLabels=[Ljava.lang.String;@190342b,
var=G2_PayloadInterface, label=Run Shell Command, method=RunShell,
paramTypes=[Ljava.lang.String;@513060, timeout=600, tooltip=Run shell command
and return result as a console ack}
2016-01-19T00:14:47.251Z@37.4108375,-122.0043515 * CFHashMap:34: *****
CFHashMap overwrite!!!
2016-01-19T00:14:47.267Z@37.4108375,-122.0043515 scripting: Add
{template=file:%trunshell N%lp, paramLabels=[Ljava.lang.String;@3efd26,
var=G2_PayloadInterface, label=Run Shell (Return File),
method=RunShellReturnFile, paramTypes=[Ljava.lang.String;@15dd533,
timeout=600, tooltip=Run shell command and return result as a file transfer}
2016-01-19T00:14:47.646Z@37.4108375,-122.0043515 * CFHashMap:34: *****
CFHashMap overwrite!!!
2016-01-19T00:14:47.653Z@37.4108375,-122.0043515 scripting: Add
{template=%tgps autostart %p, paramLabels=[Ljava.lang.String;@112d1c9,
var=G2_GPSListener, label=Set Auto Start, method=SetAutoStart,
paramTypes=[Ljava.lang.String;@8b1a0a, timeout=600, tooltip=Set whether GPS
logging starts automatically or requires manual command}
2016-01-19T00:14:47.661Z@37.4108375,-122.0043515 * CFHashMap:34: *****
CFHashMap overwrite!!!
2016-01-19T00:14:47.670Z@37.4108375,-122.0043515 scripting: Add
{template=%tgps defaultoutput %p, paramLabels=[Ljava.lang.String;@5a9737,
var=G2_GPSListener, label=Set Default Output File,
method=SetDefaultOutputFile, paramTypes=[Ljava.lang.String;@17da237,
```

```

timeout=600, tooltip=Set the GPS log file name used by default}
2016-01-19T00:14:47.679Z@37.4108375,-122.0043515 * CFHashMap:34: *****
CFHashMap overwrite!!!
2016-01-19T00:14:47.691Z@37.4108375,-122.0043515 scripting: Add
{template=%tgps output %p, paramLabels=[Ljava.lang.String;@124d0d3,
var=G2_GPSListener, label=Set Output File, method=SetOutputFile,
paramTypes=[Ljava.lang.String;@9cd3b, timeout=600, tooltip=Set the GPS log
file name to be used}
2016-01-19T00:14:48.152Z@37.4108482,-122.004358 ? SV3PayloadManager: xp meta
    java.lang.IllegalArgumentException: Must use CFHashMap!! Not
{SERVERDIR=., PHONENUM=88160000617, SERVERLOGIN=mikestestbox2@ggsmac,
KEYNAME=mikestestbox2}
    com.liquidr.regulus.config.CFHashMap.put0(CFHashMap.java:120)
    com.liquidr.regulus.config.CFHashMap.put(CFHashMap.java:43)
    java.util.HashMap.putAll(HashMap.java:622)
    com.liquidr.regulus.SV3payloads.SV3Payload.putAll(SV3Payload.java:87)
    com.liquidr.regulus.SV3payloads.SV3PayloadServlet.processRequest(SV3Payloa
dServlet.java:197)
    com.liquidr.regulus.SV3payloads.SV3PayloadServlet.doPost(SV3PayloadServlet
.java:457)
    javax.servlet.http.HttpServlet.service(HttpServlet.java:755)
    javax.servlet.http.HttpServlet.service(HttpServlet.java:848)
    org.eclipse.jetty.servlet.ServletHolder.handle(ServletHolder.java:669)
    org.eclipse.jetty.servlet.ServletHandler.doHandle(ServletHandler.java:455)
    org.eclipse.jetty.server.session.SessionHandler.doHandle(SessionHandler.ja
va:229)
    org.eclipse.jetty.server.handler.ContextHandler.doHandle(ContextHandler.ja
va:1072)
    org.eclipse.jetty.servlet.ServletHandler.doScope(ServletHandler.java:382)
    org.eclipse.jetty.server.session.SessionHandler.doScope(SessionHandler.jav
a:193)
    org.eclipse.jetty.server.handler.ContextHandler.doScope(ContextHandler.jav
a:1006)
    org.eclipse.jetty.server.handler.ScopedHandler.handle(ScopedHandler.java:1
35)
    org.eclipse.jetty.server.handler.HandlerWrapper.handle(HandlerWrapper.java
:116)
    org.eclipse.jetty.server.Server.handle(Server.java:365)
    org.eclipse.jetty.server.AbstractHttpConnection.handleRequest(AbstractHttp
Connection.java:485)
    org.eclipse.jetty.server.AbstractHttpConnection.content(AbstractHttpConnec
tion.java:937)
    org.eclipse.jetty.server.AbstractHttpConnection$RequestHandler.content(Abs
tractHttpConnection.java:998)
    org.eclipse.jetty.http.HttpParser.parseNext(HttpParser.java:856)
    org.eclipse.jetty.http.HttpParser.parseAvailable(HttpParser.java:240)
    org.eclipse.jetty.server.AsyncHttpConnection.handle(AsyncHttpConnection.ja
va:82)
    org.eclipse.jetty.io.nio.SelectChannelEndPoint.handle(SelectChannelEndPoin
t.java:628)
    org.eclipse.jetty.io.nio.SelectChannelEndPoint$1.run(SelectChannelEndPoint
.java:52)
    org.eclipse.jetty.util.thread.QueuedThreadPool.runJob(QueuedThreadPool.jav
a:608)
    org.eclipse.jetty.util.thread.QueuedThreadPool$3.run(QueuedThreadPool.java
:543)
    java.lang.Thread.run(Thread.java:722)

```

Because the registration does not complete, the payload cannot communicate with the VMC: no commands can be issued in either direction and no telemetry is reported. The apparent problem is that the new "RUDICS" section is presented as a set of name/value pairs rather than array of settings of name/value pairs.

#### Resolution

Fixed by changing the RUDICS key so that it reports as an array of maps rather than as a map:

```

2016/01/19 03:13:19.299774 dm3730_zoom_torpedo [1735] SV3NetChannel 0x20F2
PPPHost content: {
2016/01/19 03:13:19.303619 dm3730_zoom_torpedo [1735]      "RUDICS" : [
2016/01/19 03:13:19.306152 dm3730_zoom_torpedo [1735]      {

```

```

2016/01/19 03:13:19.309020 dm3730_zoom_torpedo [1735] "KEYNAME" :
"miketestbox2",
2016/01/19 03:13:19.309936 dm3730_zoom_torpedo [1735] "PHONENUM" :
"88160000617",
2016/01/19 03:13:19.310729 dm3730_zoom_torpedo [1735] "SERVERDIR" :
".",
2016/01/19 03:13:19.314880 dm3730_zoom_torpedo [1735]
"SERVERLOGIN" : "miketestbox2@ggsmac"
2016/01/19 03:13:19.315765 dm3730_zoom_torpedo [1735] }
2016/01/19 03:13:19.316802 dm3730_zoom_torpedo [1735] ],
2016/01/19 03:13:19.317626 dm3730_zoom_torpedo [1735] "boardAddress" : 0,
2016/01/19 03:13:19.318420 dm3730_zoom_torpedo [1735] "corbaName" :
"PPPHost",
2016/01/19 03:13:19.319213 dm3730_zoom_torpedo [1735] "description" :
"Host for PPP interface to internet",
2016/01/19 03:13:19.320037 dm3730_zoom_torpedo [1735] "hardware" : [
2016/01/19 03:13:19.324737 dm3730_zoom_torpedo [1735] {
2016/01/19 03:13:19.325561 dm3730_zoom_torpedo [1735] "connector" :
"_" ,
2016/01/19 03:13:19.326354 dm3730_zoom_torpedo [1735]
"manufacturer" : "Iridium",
2016/01/19 03:13:19.327148 dm3730_zoom_torpedo [1735] "model" :
"9522B",
2016/01/19 03:13:19.327972 dm3730_zoom_torpedo [1735] "serial" :
"IMEI:300025010413330 SIM:8988169414001281532",
2016/01/19 03:13:19.328765 dm3730_zoom_torpedo [1735] "tty" :
"/dev/ttyLX0"
2016/01/19 03:13:19.329559 dm3730_zoom_torpedo [1735] }
2016/01/19 03:13:19.330322 dm3730_zoom_torpedo [1735] ],
2016/01/19 03:13:19.331085 dm3730_zoom_torpedo [1735] "hostID" : "1023742
SOMDM3730-20-2780AGCR-C 2013A02943",
2016/01/19 03:13:19.332519 dm3730_zoom_torpedo [1735] "reportingAddress" :
8434,
2016/01/19 03:13:19.340393 dm3730_zoom_torpedo [1735] "tag" : "PPPHost",
2016/01/19 03:13:19.341217 dm3730_zoom_torpedo [1735] "ui" : [
2016/01/19 03:13:19.341979 dm3730_zoom_torpedo [1735] {
2016/01/19 03:13:19.342773 dm3730_zoom_torpedo [1735] "label" :
"Call Now",
2016/01/19 03:13:19.343566 dm3730_zoom_torpedo [1735] "method" :
"CallNow",
2016/01/19 03:13:19.344329 dm3730_zoom_torpedo [1735] "template" :
"%tCALLNOW",
2016/01/19 03:13:19.345123 dm3730_zoom_torpedo [1735] "timeout" :
600,
2016/01/19 03:13:19.345947 dm3730_zoom_torpedo [1735] "tooltip" :
"Attempt to make a RUDICS connection now (and perform default actions)"
2016/01/19 03:13:19.346771 dm3730_zoom_torpedo [1735] },
2016/01/19 03:13:19.347534 dm3730_zoom_torpedo [1735] {
2016/01/19 03:13:19.348541 dm3730_zoom_torpedo [1735] "label" : "Get
Key Name",
2016/01/19 03:13:19.349365 dm3730_zoom_torpedo [1735] "method" :
"GetKeyName",
2016/01/19 03:13:19.350158 dm3730_zoom_torpedo [1735] "template" :
"%tGETKEY",
2016/01/19 03:13:19.350952 dm3730_zoom_torpedo [1735] "timeout" :
600,
2016/01/19 03:13:19.351745 dm3730_zoom_torpedo [1735] "tooltip" :
"Reads the configured key name"
2016/01/19 03:13:19.355987 dm3730_zoom_torpedo [1735] }
2016/01/19 03:13:19.356781 dm3730_zoom_torpedo [1735] ]
2016/01/19 03:13:19.357543 dm3730_zoom_torpedo [1735] }
2016/01/19 03:13:19.358551 dm3730_zoom_torpedo [1735] HTTP POST
http://192.168.105.1/xp?op=meta&id=20F2

```

PPPHost now also has a configuration value (includeRUDICSDetailInRegistration) that indicates whether to include RUDICS key. Regulus 1.0.3 is able to accept this configuration, and it still works on Regulus 1.0.2

## Dependencies

This version of software requires RegulusOS 1.1.1 (P/N 05858-02) and was tested with Regulus 1.0.2 and 1.0.3

## Open Issues

SWSMAC-343 "MAM reports a non-zero peak wave direction in sensor telemetry when reporting an invalid sample (i.e. with significant wave height, peak period and average period set to zero.)"

SWSMAC-445 "MOSE did not stop sampling with MOSE --off and MOSE --stop, even though acks were returned"

SWSMAC-460 "Implement communication channels over the expansion port interface"

SWSMAC-461 "Enable expansion port clients to receive copies of telemetry messages"

SWSMAC-462 "Enable expansion port clients to receive copies of VMC-resident low-level feeds"

SWSMAC-463 "Re-implement startup support for encrypted partitions using either expansion port or sensor port"

## End-user Impacts

When a user upgrades to Regulus 1.0.3, SMC version 2.5.0 with a PPPHost (payload based RUDICS modem) will fail to register with Regulus. A user with any SMC version 2.5.0 with RUDICS will be required to upgrade to 2.5.1 if using Regulus 1.0.3.

A user with Regulus 1.0.2 may use SMC version 2.5.1 without any known issues.

## Support Impacts

Software installation bundles are now distributed using a self-extracting script: the upgrade and installation process has been simplified since the script can inspect the environment, save important files, unpack the software, restore saved files and prompt the user to set program options in a more friendly way.

The release will include a binary image of the SD card with multiple part numbers installed: one with ADCP configurations and one without. This simplifies creation of a *golden master* or to completely overwrite a used micro-SD card.

## Installation/Upgrade Instructions

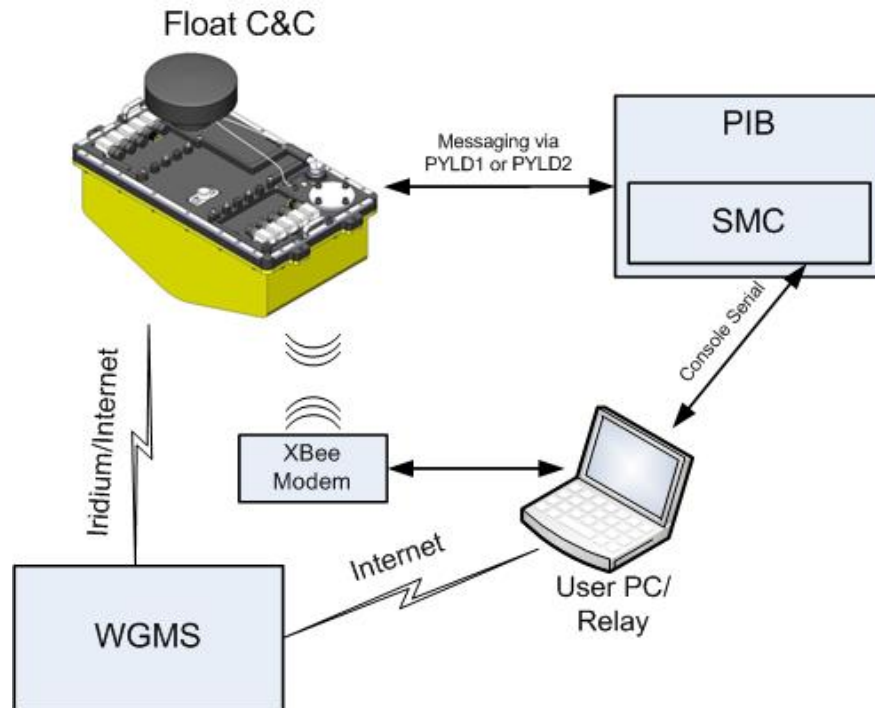
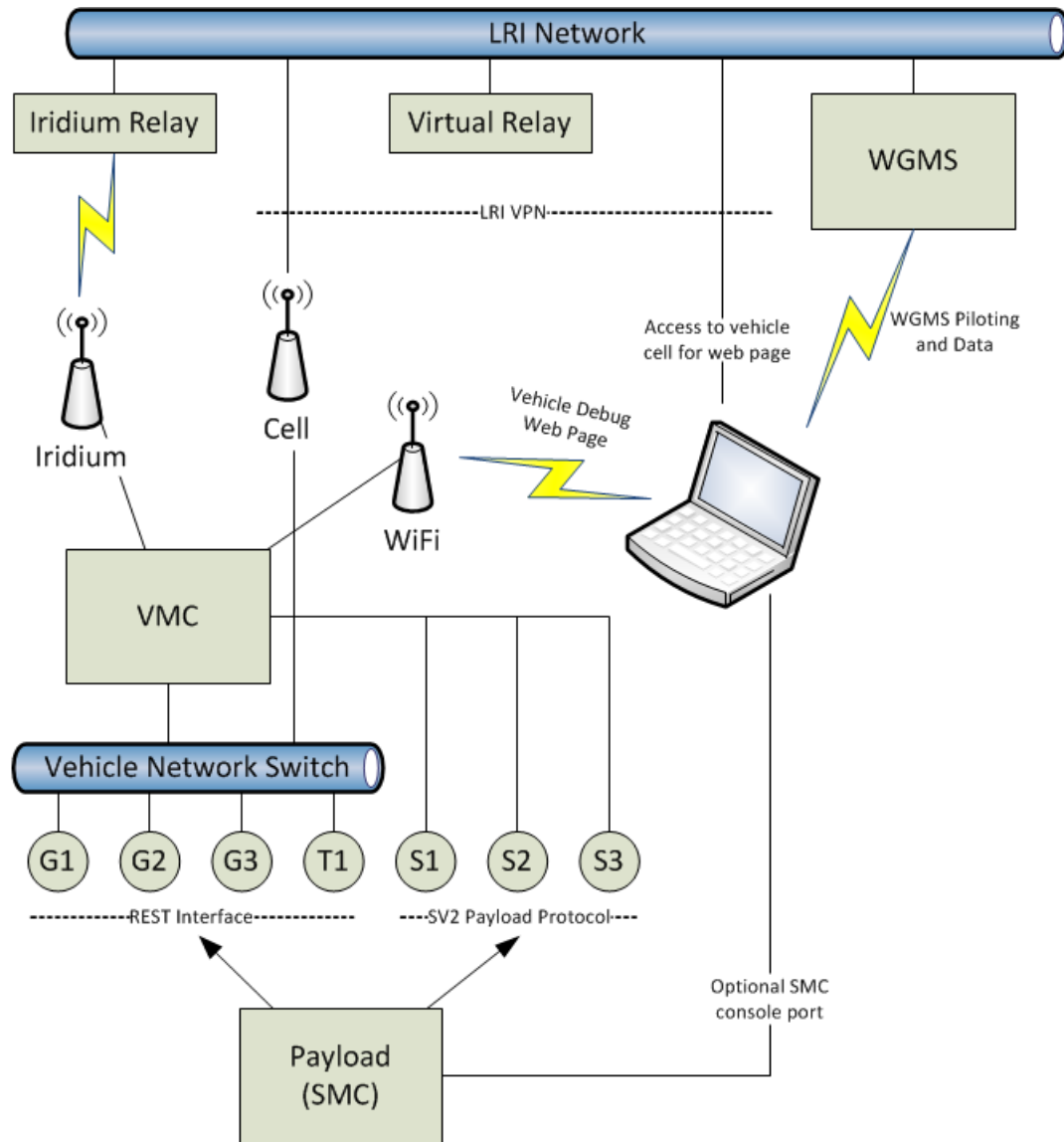


**Figure 1 SV2 SMC interconnection with Float C&C and user PC (setup configuration)**

WGMS (Wave Glider Management System) is a web-based application and database that provides a command and telemetry interface to vehicles. It communicates to vehicles via the Iridium network, but extends its reach via software relays.

The Float C&C is the “command and control” computer that supervises navigation and acts as a message router for commands and data for the vessel. It also implements the Iridium communication channel for access at sea, and allows for low-cost short range communication via XBee for development, setup and troubleshooting. The XBee relay (P/N 00775-XX) also exposes a terminal interface for debug message display and entering low-level commands.

The SMC implements additional communication channels, but for this setup case, communication will be via the Float C&C and XBee or via the SMC’s USB-based console.



### Figure 2 SV3 SMC interconnection with Float C&C and user PC

In the SV3 configuration, the SMC can interface with the VMC (Vehicle Management Computer, the equivalent of the SV2 Float C&C) in the same way that it does on SV2 using Payload Protocol over a Sensor Port; but SV3 Sensor Ports don't include umbilical aux lines, so it's not possible to talk to devices attached to the Glider or in a towfish.

It can also be connected via an Expansion Port, which provides Ethernet to the payload. The VMC exposes a REST interface with same kind of operations available through Payload Protocol. Expansion Ports carry programmable spare lines that can be used to carry GPS/PPS or umbilical aux lines.

Expansion ports also carry AMPS communication lines. AMPS is dedicated to controlling and reporting on the power components: generators, batteries, consumers, power supplies and switches. SV3 payload boxes may have power switches that switch unregulated  $V_{\text{consumer}}$  power or regulators selected for the

particular application. Currently, AMPS components can only be interfaced from the VMC or via its interfaces.

Note that SV3 does not implement an XBee radio. Instead, it implements a web page with diagnostics and setup capabilities accessible via WiFi or the cell modem. Use of the cell modem requires VPN credentials.

### Setting up the SD Card

The SD card part (02574-XX) is created by setting up 3 partitions: BOOT, LINUX and XDATA:

- The BOOT partition contains instructions to the Logic Loader software (which is programmed on the CPU) as well as U-Boot and the Linux kernel.
- The LINUX partition contains the Linux root filesystem and any unencrypted SMC application files.
- The XDATA partition contains log files and resulting SMC data. XDATA can be configured as an encrypted filesystem, in which case the majority of the application is also loaded on XDATA.

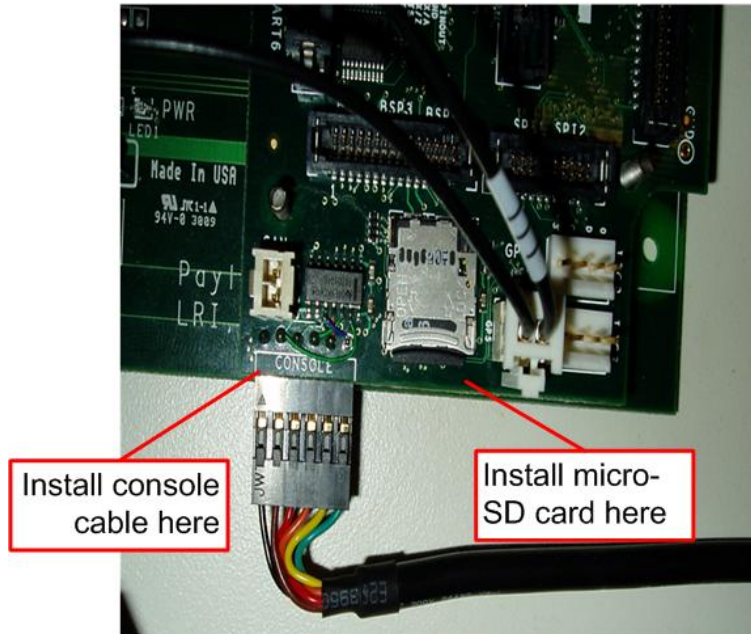
The decision of whether to encrypt the XDATA partition is typically made when creating the SD card. For systems requiring the security of an encrypted partition, care should be taken not to store anything sensitive – even temporarily – on the unencrypted (LINUX) partition.

### Working with the SMC and the SV2 Float C&C

The PIB (Payload Interface Board) is a general purpose controller used to implement protocols to sensors or other computers. It has three (3) interface ports with their own power switches and serial ports. Special purpose interface boards (PCOMM's) are installed depending on the hardware protocol for each port: RS-232, RS-422 or RS-485.

The SMC can be mounted in place of one of the interface boards (two ports are consumed because of the size of the SMC.) To mount a SMC:

1. Ensure that the Float C&C payload port is powered off. This can be accomplished via the XBee Relay's debug console using command #29 (PYLD1) or #31 (PYLD2) or by issuing commands via WGMS.
2. Disconnect the PIB from the payload port (PYLD1 or PYLD2)
3. Install the SMC on PIB port J2; the PIB must be loaded with software version 2.0.335 (P/N 02639-01) or greater.
4. Reconnect the PIB to the appropriate payload port.



5. With an open payload box or a standalone SMC board, interconnect the SMC with a PC using the USB console cable (P/N 02638). When using a payload box that includes an external debug port, use an external debug cable (P/N 02824).
6. Ensure that a micro-SD card with the correct kernel version has been installed in the SMC.
7. Open the USB serial port provided by the console cable at 115,200 baud using a serial terminal like HyperTerminal or putty (Windows) or GkTerm, minicom or screen (Linux).
8. From WGMS, use the Set Parameter command to change the Payload Power On Enumeration Delay appropriately (120 seconds recommended.) This enables the SMC software to start up in time to respond to the Float C&C's enumeration request.
9. Enable power to the payload from the XBee Relay's debug console using command #28 (PYLD1) or #30 (PYLD2) or by issuing commands from WGMS. Momentarily, the LED's on the board turn on and messages from the Linux startup process begin to appear from the SMC console port. Finally, a Linux login prompt appears.
10. Watch the XBee Console for the response to the enumeration request. It should look something like this:

```
Description: Full PIB
Address: 0100 Major: 2 Minor: 1 Build: 368
Payload 1 Port

Description: SMC/Encrypted
Address: 2000 Major: 1 Minor: 1 Build: 5
Payload 1 Port
```

If you don't see a response from the SMC, one of the following may be true:

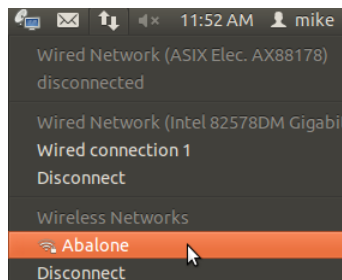
- There is no SMC software loaded.

- The PIB has not been configured to talk to the SMC at that particular address. See the user manual for the PIB for instructions on how to configure it.
- The wrong type or version of PIB software has been installed. In this case, copy the enumeration response and send it to LRI customer support for advice.
- The enumeration timeout is too short. Issuing the power on command again without powering it off first will restart the enumeration process; if the SMC then responds, try increasing the power on enumeration delay.

Note that it is important for the SMC to respond since the Float C&C uses enumeration responses to determine which addresses to poll. If the SMC's address is not polled, it cannot send any messages to the Float C&C.

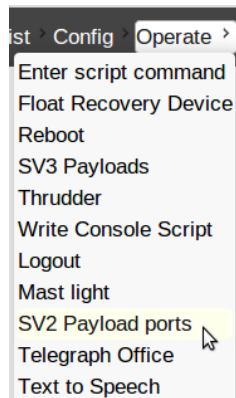
### Working with the SMC and the SV3 via Sensor Port

SV2 payloads work similarly on SV3 to how they worked on SV2, but the diagnostics and operations can be performed via the vehicle's web page. The vehicle acts as a WiFi server with a SSID similar to the vehicle name. My test vehicle is called Abalone.



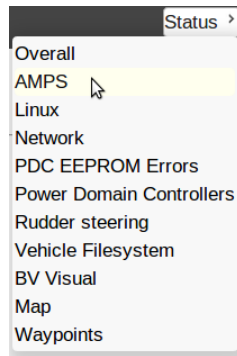
WiFi clients are assigned an IP address in the 192.168.100.X domain; the vehicle can be reached at 192.168.100.1. You should have received login credentials with your vehicle; otherwise, call Customer Support for a web page login.

The opening page shows the status of various services and devices. Pull down the "Operate" menu and select "SV2 Payload ports":





To open the AMPS page, pull down the “Status” menu and select “AMPS”:



Scroll down to the “Configure Expansion Switches” link (typically under slot #34) and click the link:

- **Slot 34:** Expansion Switch Module  
Firmware Version: 0x000035f1, Boot Loader Version: 0x0000381e  
Mfg. S/N=abalone434  
HW ID=0x002c001b3533363305473131  
Reported Arena Part Number = 4572A (matches manifest)  
Module Type = 7  
Found in Manifest by Serial Number.  
[Configure Expansion Switches](#)
- #1: Umbilical Expansion Switch: Not Connected
- #2: Debug Expansion Switch: Not Connected
- #3: G1 Expansion Switch: Connected to GPS/PPS Signals
- #4: G2 Expansion Switch: Connected to GPS/PPS Signals
- #5: G3 Expansion Switch: Connected to GPS/PPS Signals

The switch configuration page is displayed; make any desired changes in mapping and then click the “Configure Expansion Switches” button.

#### Expansion Switch Configuration

When configuring the Aux Lines for the expansion ports, the options available will depend on the port in question. At this point, Any (and ALL) expansion ports may use the GPS/PPS option regardless of the configuration of other ports. The AUX lines from only one expansion port may be connected to the Umbilical at any time (If G1 is using that option; then G2, and G3 CAN NOT be configured use it) If any of these conditions are not true, then the settings will not be updated.

**Umbilical Aux Line Settings**

☒ Aux Lines Are Not Used

☐ Someone (G1, G2, G3, or Debug) is Connecting to Me

☐ Aux Lines Connect To GPS and PPS signals

☐ Aux Lines Connect To the Serial Port on the VMC

**G1 Aux Line Settings**

☐ Aux Lines Are Not Used

☒ Aux Lines Connect To GPS and PPS signals

☐ Aux Lines Connect To The Umbilical

**G2 Aux Line Settings**

☐ Aux Lines Are Not Used

☒ Aux Lines Connect To GPS and PPS signals

☐ Aux Lines Connect To The Umbilical

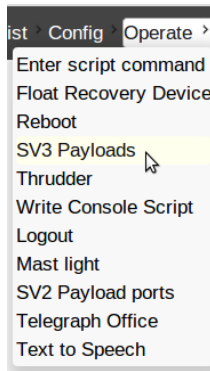
**G3 Aux Line Settings**

☐ Aux Lines Are Not Used

☒ Aux Lines Connect To GPS and PPS signals

☐ Aux Lines Connect To The Umbilical/Towbody

Once the setting is correct, pull down the “Operate” menu and select “SV3 Payloads”



The SV3 Payloads page has a section for each expansion port:

- When using a payload box that contains an AMPS domain, click the “Turn On SV3 Integrator Payload” button. This not only powers on the payload box, but also looks for the AMPS domain and determines if specific power switches should be turned on. For example, is an SMC known to be configured on power switch #2? If so, the appropriate switches are set.
- If no AMPS domain is in the box, click the “Turn On Port” button to apply power.
- When turning off the payload, click “Unplug Integrator Payload” to not only turn off power but also forget the registration data (in preparation for disconnecting the payload box from the VMC.)
- To simply turn the power off, click “Turn Off Port.” Note that either power off button gives smart payloads the opportunity to shut down gracefully.

|                 |               |                                |                           |
|-----------------|---------------|--------------------------------|---------------------------|
| Refresh         |               |                                |                           |
| Expansion Ports |               |                                |                           |
| T1 is On        | Turn Off Port | Unplug Integrator Payload      |                           |
| G2 is On        | Turn Off Port | Unplug Integrator Payload      |                           |
| G3 is Off       | Turn On Port  | Turn On SV3 Integrator Payload | Unplug Integrator Payload |
| G0 is Off       | Turn On Port  | Turn On SV3 Integrator Payload | Unplug Integrator Payload |
| G1 is Off       | Turn On Port  | Turn On SV3 Integrator Payload | Unplug Integrator Payload |

If one or more payloads are registered using the expansion port REST interface, the detail is displayed graphically.

|                                                                                                                                                                                                                                      |                                                                                                                                                                             |                                                                                                                                                                                                         |                                                                                                                                                                                                                                |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| G2_PayloadInterface<br>192_168_105_103_2000                                                                                                                                                                                          |                                                                                                                                                                             |                                                                                                                                                                                                         |                                                                                                                                                                                                                                |
| Controlled by G2 On                                                                                                                                                                                                                  |                                                                                                                                                                             |                                                                                                                                                                                                         |                                                                                                                                                                                                                                |
| ui                                                                                                                                                                                                                                   | <pre>template %stgetmodules method GetModules label Get Modules timeout 600 tooltip Report the configured modules running on the SMC</pre>                                  | <pre>template %strunshell %p paramLabels [Command] method RunShell label Run Shell Command paramTypes [java.lang.String] timeout 600 tooltip Run shell command and return result as a console ack</pre> | <pre>template file:%strunshell %p paramLabels [Command] method RunShellReturnFile label Run Shell (Return File) paramTypes [java.lang.String] timeout 600 tooltip Run shell command and return result as a file transfer</pre> |
| <pre>reportingAddress 892 description Payload interface tag G2_PayloadInterface localTag PayloadInterface boardAddress 0 corbaName PayloadInterface hostID 1023742 SOMDM3730-20-2780AGCR-C 2013A02943 interface [server 1.4.0]</pre> |                                                                                                                                                                             |                                                                                                                                                                                                         |                                                                                                                                                                                                                                |
| Configuration Parameters                                                                                                                                                                                                             |                                                                                                                                                                             |                                                                                                                                                                                                         |                                                                                                                                                                                                                                |
| ui                                                                                                                                                                                                                                   | <pre>template %stgetmodules method GetModules label Get Modules timeout 600 tooltip Report the configured modules running on the SMC</pre>                                  | <pre>template %strunshell %p paramLabels [Command] method RunShell label Run Shell Command paramTypes [java.lang.String] timeout 600 tooltip Run shell command and return result as a console ack</pre> | <pre>template file:%strunshell %p paramLabels [Command] method RunShellReturnFile label Run Shell (Return File) paramTypes [java.lang.String] timeout 600 tooltip Run shell command and return result as a file transfer</pre> |
| <pre>reportingAddress 892 localTag PayloadInterface tag G2_PayloadInterface description Payload interface boardAddress 0 corbaName PayloadInterface hostID 1023742 SOMDM3730-20-2780AGCR-C 2013A02943 interface [server 1.4.0]</pre> |                                                                                                                                                                             |                                                                                                                                                                                                         |                                                                                                                                                                                                                                |
| G2_GPSListener<br>192_168_105_103_2001                                                                                                                                                                                               |                                                                                                                                                                             |                                                                                                                                                                                                         |                                                                                                                                                                                                                                |
| Controlled by G2 On                                                                                                                                                                                                                  |                                                                                                                                                                             |                                                                                                                                                                                                         |                                                                                                                                                                                                                                |
| ui                                                                                                                                                                                                                                   | <pre>template %stgps autostart method GetAutoStart label Get Auto Start timeout 30 tooltip Determine if GPS logging starts automatically or requires a manual command</pre> | <pre>template %stgps defaultoutput method GetDefaultOutputFile label Get Default Output File timeout 30 tooltip Get the GPS log file name used by default</pre>                                         | <pre>template %stgps output method GetOutputFile label Get Output File timeout 30 tooltip Get the GPS log file name currently in use</pre>                                                                                     |
| <pre>template %stgps autostart %p paramLabels [New State] method SetAutoStart label Set Auto Start paramTypes [java.lang.String] timeout 30 tooltip Set whether GPS logging starts automatically or requires manual command</pre>    |                                                                                                                                                                             |                                                                                                                                                                                                         |                                                                                                                                                                                                                                |
| <pre>template %stgps start method StartGPSLogging label Start GPS Logging timeout 30 tooltip Start GPS logging to a file</pre>                                                                                                       |                                                                                                                                                                             |                                                                                                                                                                                                         |                                                                                                                                                                                                                                |
| <pre>template %stgps stop method StopGPSLogging label Stop GPS Logging timeout 30 tooltip Stop GPS logging to a file</pre>                                                                                                           |                                                                                                                                                                             |                                                                                                                                                                                                         |                                                                                                                                                                                                                                |
| reportingAddress 8433                                                                                                                                                                                                                |                                                                                                                                                                             |                                                                                                                                                                                                         |                                                                                                                                                                                                                                |

## Installing the Application

- Before starting, put any software archives to be installed in a convenient place to be accessed. With the SD card inserted into a card reader and connected to Ubuntu:

- For systems without an encrypted filesystem, copy the archives to a convenient location like /root

```
sudo cp tarball-filename /media/LINUX/root
sudo umount /media/*
```

- For systems using an encrypted filesystem, the archives should never be stored on the SD card unencrypted. In this case, mount the encrypted filesystem and copy the archive to the mount point

```
echo 'passphrase' | sudo cryptsetup create --cipher aes-
cbc-essiv:sha256 --key-size 256 --hash sha256 secsd
/dev/sdd3
sudo mount /dev/mapper/secsd /mnt/secsd
sudo cp tarball-filename /media/mapper/secsd
sudo umount /mnt/secsd
sudo cryptsetup remove secsd
sudo umount /media/*
```

Note that the micro SD card device (/dev/sdd in my example) will vary depending on what other devices you have installed.

Be sure to wait for the umount command to complete before removing the micro SD card.

2. Insert the micro SD card into the SMC and power it up.
3. Log in as root
4. Set the password for the root account using **passwd**
5. Create the **smc** account (**useradd smc**) and enable access to serial ports (**usermod -a -G dialout smc**)
6. Set the password for the **smc** account (**passwd smc**)
7. Create the **smc0** account (**useradd smc0**), enable access to serial ports (**usermod -a -G dialout smc0**) and set the password (**passwd smc0**)
8. For systems using encryption, mount the XDATA partition under /home/smc:

```
echo 'passphrase' | cryptsetup create --cipher aes-cbc-essiv:sha256 --key-size 256 --hash sha256 secsd /dev/mmcblk0p3

mount /dev/mapper/secsd /home/smc
```

9. Install the application archive(s):

```
tar xvpf /path-to-archive/tarball-filename -C /
```

10. Exit the **root** user and log in as **smc**

11. For systems that do not use filesystem encryption, mount the XDATA partition and set it up:

```
sudo mount /dev/mmcblk0p3 /mnt/XDATA
sudo bin/SetupXDATA
```

12. The startup script (/home/smc/startup) configures the transceivers for some of the serial ports for different hardware protocols. Change the line that with "InitOctalUartLines" to match the desired serial port configuration of ports /dev/ttyLX2-/dev/ttyLX7. For each port, enter 'rs232', 'rs422' or 'off'; there should be a total of six settings. For example, the following enable rs232 on /dev/ttyLX2 and /dev/ttyLX3, but /dev/ttyLX4-7 are not disabled:

```
sudo InitOctalUartLines rs232 rs232 off off off off
```

13. The /home/smc/roots directory contains a sub-directory for each application instance to be executed. By default, it contains one entry for each application, so the content must be customized for each shipping configuration.

```
smc@omap35x_torpedo:~/roots$ ls -l
total 76
drwxr-xr-x 2 smc smc 4096 Jan  1  2000 001_omniNames
drwxr-xr-x 2 smc smc 4096 Jan  1  2000 002_EventManager
drwxr-xr-x 2 smc smc 4096 Jan  1  2000 050_PayloadManager
drwxr-xr-x 2 smc smc 4096 Jan  1  2000 100_PayloadInterface
drwxr-xr-x 2 smc smc 4096 Aug 23 01:09 190_BenthosModem
drwxr-xr-x 2 smc smc 4096 Jan  1  2000 200_GPSListener
drwxr-xr-x 2 smc smc 4096 Jan  1  2000 200_MAM
drwxr-xr-x 4 smc smc 4096 Aug 23 03:18 200_PPPHost
-rw-r--r-- 1 smc smc 1746 Jan  1  2000 AddressSummary.txt
```

*Sample listing of configuration directories under /home/smc/roots*

The applications are run in sorted order, and there is a three-digit prefix that is used by convention to establish run groupings:

- 001 is reserved for omniORB, which must run first so that applications can use CORBA services.
- 002 is reserved for EventManager, which must be available for applications that send or receive events.
- 050 is reserved for PayloadManager, which must be available for applications that register an enumerable name; since PayloadManager fires events to indicate when new names are registered, it must be after EventManager.

Other similar services can be registered in the 50-99 range.

- 100 is used by PayloadInterface, which implements the communication protocol with the Float C&C. Since some applications make use of PayloadInterface services, it makes sense for it to be available before they start.

Other similar services can be registered in the 100-199 range.

- 200 is used by most other applications. Some of these probably will not be used; services like GPSTListener (which creates events from the GPS receiver) or PPPHost (which offers communication via RUDICS) will likely be required, depending on the hardware installed on the system.

By default, component directories contain a ".disabled" file that prevents them from starting. Delete this file to enable applications to start up. If you are installing a preconfigured archive, this will have already been done.

14. Additional component configuration steps are required for some applications (see below) The **startup** script loads the octal UART driver; it may be necessary to execute it and then run **sudo killsmc** to

15. Restart the system to run the default scenario (`sudo shutdown -r now`)

## Changing Encryption Passphrases

Filesystem encryption works by performing all reads and writes using the passphrase provided: the data on the storage medium is actually changed. It's not like a password that lets you in the door: it's more like a filter. Therefore, changing the passphrase is easy. It means reading the entire partition using the old passphrase to decrypt it, reformatting the partition using the new passphrase and then rewriting the partition data. For example:

1. Insert the micro SD card into to a reader connected to Ubuntu
2. Mount the old partition

```
echo 'old-passphrase' | sudo cryptsetup create --cipher
aes-cbc-essiv:sha256 --key-size 256 --hash sha256 secsd
/dev/sdd3

sudo mount /dev/mapper/secsd /mnt/secsd
```

3. Copy the partition data to a location elsewhere

```
pushd /mnt/secsd

sudo tar cvf /path-to-other-location/save-archive .
popd
```

4. Unmount the partition

```
sudo umount /mnt/secsd  
sudo cryptsetup remove secsd
```

5. Reformat the partition

```
echo 'new-passphrase' | sudo cryptsetup create --cipher  
aes-cbc-essiv:sha256 --key-size 256 --hash sha256 secsd  
/dev/sdd3  
  
sudo mkfs.ext3 -L XDATA /dev/mapper/secsd  
  
sudo tune2fs -c 0 /dev/mapper/secsd
```

6. Re-mount the partition

```
sudo mount /dev/mapper/secsd /mnt/secsd
```

7. Copy the partition data back:

```
pushd /mnt/secsd  
  
sudo tar xvf /path-to-other-location/save-archive  
  
popd
```

8. Unmount the partition again

```
sudo umount /mnt/secsd  
sudo cryptsetup remove secsd
```

## Component Configuration

Each component or application instance to be started automatically has a corresponding subdirectory under `/home/smc/roots`. These directories include:

- A **startup** script that manages the startup process for the component.
- One or more XML-based configuration files specific to the executable.
- Other scripts used by the application.
- Data sub-directories or links to directories where data is stored.

The **startup** script is typically responsible for:

- Performing any initial cleanup from previous application runs.
- Executing the application, including any parameters to be passed.
- Setting up logging for applications that produce logs on standard output.
- Adding entries to the smc user's crontab to allow for cron-based callbacks.
- Performing any initial commands to configure behavior.

The XML configuration file (typically named *application-config.xml*) contains various settings like:

- Serial port name and parameters for serial-based sensors.
- CORBA parameters, like the name that the application will register and the names of other applications with which it will interact.
- Command verbs (what shell command to execute in a particular circumstance, like powering a sensor on or off)
- Event names for events that will be produced and ones that the application is looking for.
- Timing and retry limits

Most of these parameters will work without modification, but some exceptions are:

- Serial port names and references to electrical switches (these should be checked vs. the wiring of your system)
- The board address associated with each process, if the SMC is not configured to report as address 2000 (hex)
- References to external systems and directories that may vary by Customer or which must be unique per vehicle
- Different use cases that affect timing values
- Event naming: in one case, component A produces "Event1" and component B processes it; in another case, component A produces "Event1" that component C processes and merges with other data and produces "Event2", which component B consumes.
- Configuration of services used in common by multiple applications.

Each component can also have a JSON registration file that is rolled up into the manifest file. The JSON file should be named according to the CORBA server name; for example, the Payload Interface process has a CORBA server name of "PayloadInterface," so its JSON description file is "PayloadInterface.json"

The content of JSON file, at a minimum, the list of commands that are exposed by the process, argument names and types, tool tip descriptions and a string template that describes how the arguments will be formatted if the command is invoked. For example, the following is the JSON published by Payload Interface:

```
{
  "ui" : [
    {
      "label" : "Get Modules",
      "method" : "GetModules",
      "template" : "%tgetmodules",
      "timeout" : 600,
      "tooltip" : "Report the configured modules running on the SMC"
    },
    {
      "label" : "Run Shell Command",
      "method" : "RunShell",
      "paramLabels" : [ "Command" ],
      "paramTypes" : [ "java.lang.String" ],
      "template" : "%trunshell %p",
      "timeout" : 600,
      "tooltip" : "Run shell command and return result as a console ack"
    },
    {
      "label" : "Run Shell (Return File)",
      "method" : "RunShellReturnFile",
      "paramLabels" : [ "Command" ],
      "paramTypes" : [ "java.lang.String" ],
      "template" : "file:%trunshell %p",
      "timeout" : 600,
      "tooltip" : "Run shell command and return result as a file transfer"
    }
  ]
}
```

In the above:

- *label* is a text string to put on a button on the UI;
- *method* is the name sent back to the VMC when the command is invoked (the method must not include spaces);
- *paramLabels* is an array of strings describing each parameter to the command;

- *paramTypes* is an array of strings that describe the type of each command parameter, for example:
  - "boolean" is a logical value (true or false);
  - "int" is a 32-bit signed integer;
  - "float" is a single-precision IEEE 754 floating point value;
  - "double" is a double-precision IEEE 754 float point value;
  - "java.lang.String" is a sequence of characters;
- *template* indicates how to convert the parameters to a command string when sending it back to the payload. The string can contain printf-style format sequences that start with a percent sign ("%"); two percent signs ("%") will expand to a single percent sign. The following format sequences are defined:

|        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| %m     | The method name                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| %nnffp | Parameters with optional formatters. If a number (nn) is specified, it indicates which parameter to expand; otherwise, all parameters are expanded. Other format character (ff) are as follows: <ul style="list-style-type: none"><li>* Use spaces to separate parameters (default)</li><li>, Use commas to separate parameters</li><li>C Quote parameters using C language standards (defaults to shell rules)</li><li>U Quote parameters using URL rules (defaults to shell rules)</li></ul> |
| %fft   | The command tag that must be passed back to reply to the command with optional formatters                                                                                                                                                                                                                                                                                                                                                                                                      |

- *tooltip* is a string to give more information about the command when the mouse hovers over the button (or whenever it would be appropriate.)

See the SV3 ICD for additional detail.

## Aquatracka (Chelsea Technologies Group) Configuration

Aquatracka implements a CORBA command and event interface to a Chelsea Aquatracka fluorimeter. This section indicates what has changed and gives some setup guidance, but please consult the LRI Aquatracka Sensor User's Guide for additional detail.

### Commands

Commands to Aquatracka are sent to process name "Aquatracka" with ServerExec or it is typically assigned an address of 0x2016 (the upper byte varies with the base address of the SMC.) The available commands are:

AQUAT --start

Wakes up the sensor and begins streaming data.

AQUAT --stop

Stops the sensor from streaming data (sensor is still powered on)

AQUAT --run *command-file*

Runs a set of commands in the given command file.

AQUAT [-o | --output] *output-file*

Directs raw output from the sensor to the given output file.

AQUAT --off

Switches off power to the sensor.

AQUAT --on

Switches on power to the sensor.

AQUAT --getconfig

Reads the configuration from the sensor.

AQUAT --header

Reads sensor header information (serial number and configuration constants.)

AQUAT --pause

In data collection mode, halts reporting of sensor data pending a resume command (power draw remains at data acquisition level.)

AQUAT --resume

In data collection paused mode, resumes reporting of sensor data.

AQUAT --verbose

Places the sensor in verbose reporting mode.

AQUAT --noverbose

Places the sensor in non-verbose reporting mode.

AQUAT --temperature

Polls for the temperature, which is reported in degrees (C) The sensor must have already been started.

AQUAT --concentration

Polls the Fluorophor concentration (gm/cc); the sensor must have already been started.

AQUAT --calibtemp

Turns off power to the sensor by executing the power off verb from the XML file.

AQUAT -calibref *n1 n2 n3*

Writes calibration reference values into Calibration-config.xml

AQUAT -calibsig *n1 n2 n3*

Writes signal calibration constants into Calibration-config.xml

AQUAT -calibserial *newvalue*

Writes sensor serial number into Calibration-config.xml

AQUAT -calibtemp *n1 n2 n3*

Writes temperature calibration constants into Calibration-config.xml

SELECT [ -t | --time | -s | --samples | -d | --distance ] *N*

Configures the way that samples are selected for export (see EXPORT command)

EXPORT [ -count ] [ -o | --output *output-filename* ]

Configures the name of the file where selected samples are written.

## BenthosModem Configuration (not supported)

*Note that the BenthosModem module is not a supported service.*

BenthosModem implements a command interface and a pass-through data pseudo-port for use with Benthos acoustic modems. There are at least three use cases covered:

1. Acoustic range detection and communication test: this is initiated by the 'BAM START-RANGING *target-address ping-interval*' command (terminated by 'BAM STOP-RANGING') and results are reported via telemetry. This operation may be needed to locate the sea floor modem and calibrate communication parameters.
2. Pass-through mode: a client application sends and receives data via the local modem and the acoustic link to device connected to a remote (sea floor) modem. Data to be sent to the remote device are written to a pseudo port; data received from the remote device can be read from a pseudo port.
3. Collection of recorded data: a client program or script issues commands like 'BAM READDATARECORDER *target-address block-range*' to retrieve data stored in a remote (sea floor) modem's data recorder; this data has been passively-received and stored by a sea floor modem operating in low power mode.

## Configuring the Client Program

For clients that operate in pass-through mode, verify that the pseudo-port name (typically /home/smc/roots/190\_BenthosModem/pseudotty) is used in place of a serial port. In addition, the client program will need to invoke commands (for example, 'BAM CONNECT *target-address*') on the BenthosModem process in order to establish connection.

## Loading the Modem Firmware

The BenthosModem process was written and tested with modem firmware version 8.2.2. Note that sea floor modems using the older generation of firmware (6.x.x) are incompatible with later generations (8.x.x), so the set should be upgraded together.

To read the current version of installed firmware:

1. Connect the modem serial port to an available PC serial port or verify connection to the SMC

2. Open the USB serial port provided by the console cable at 9600 baud using a serial terminal like HyperTerminal or putty (Windows) or GkTerm, minicom or screen (Linux)
3. Disable hardware handshaking on the terminal program
4. Enable power to the modem; for example, on the SMC: `sudo SetPowerSwitch 2 1` (modify the switch number as necessary)
5. Wait for the chirps from the modem to indicate that it has powered up; a will be displayed indicating the current firmware version:

```
Teledyne Benthos ATM-900 Series
LF Frequency Band
Standard version 8.2.2
Aug 18 2011 19:26:49
```

Configuration BOM P/N 02690 lists the modem firmware upgrade kits that are available to use, or contact Teledyne Benthos for instructions and additional firmware.

### Configuring Modem Parameters

Once the modem's firmware is set, the operational parameters should be set as follows:

1. Connect the modem serial port to an available PC serial port or verify connection to the SMC
2. Open the USB serial port provided by the console cable at 9600 baud using a serial terminal like HyperTerminal or putty (Windows) or GkTerm, minicom or screen (Linux)
3. Disable hardware handshaking on the terminal program
4. Enable power to the modem; for example, on the SMC: `sudo SetPowerSwitch 2 1` (modify the switch number as necessary)
5. After the startup banner is displayed, the modem either responds with a command prompt or a 'CONNECT' message. 'CONNECT' indicates that modem is in pass-through mode: any characters typed will be transmitted. To get back into command mode, type +++ (three plus signs) quickly.
6. At the command prompt, issue the following commands for a top side modem:

```
>AT&F
Factory Reset
user:5>@prompt=1
Prompt | 1
>ATS8=2
OK
>ATS10=255
OK
>ATS13=3
OK
>ATS15=0
OK
>@strictat=ena
Entering strictat mode
```

```

StrictAT          | Ena
>AT&W
Sregisters Stored
>

```

For sea floor modems, use the following script:

```

>AT&F
Factory Reset
user:5>@prompt=1
Prompt          | 1
>ATS13=0
OK
>ATS15=1
OK                                     (startup mode: 1=online, 2=data logger)
>ATS18=nnn
OK                                     (nnn = the address of the sea floor modem)
>@strictat=ena
Entering strictat mode
StrictAT          | Ena
>AT&W
Sregisters Stored
>

```

## C3 (Turner Designs) Configuration

C3 is a command and event interface to a Turner C3 fluorometer; this is a re-implementation of the PIB-based C3 interface, and the commands are kept similar for continuity.

### Commands

Commands to C3 are sent to process name "C3" with ServerExec or it is typically assigned an address of 0x2023 (the upper byte varies with the base address of the SMC.) The available commands are:

```

C3 -o
    Turns on the power switch associated with the sensor

C3 -f
    Turns off the power switch associated with the sensor

C3 -r
    Alias for C3 -o (turns on the power switch associated with the sensor)

C3 -x
    Alias for C3 -f (turns off the power switch associated with the sensor)

```

### Events and Telemetry

The C3Event type contains the following name-value pair items:

```

"C3EventName" = C3Event_C3SampleHeader or C3Event_C3SampleData
"Header" = header string received from sensor
"Date" = date value (format mm/dd/yyyy) in the sample data received from sensor
"Time" = time value (format hh:mm:ss) in the sample data received from sensor
"DataC1" = first data item in the sample data from sensor
"DataC2" = second data item in the sample data from sensor

```

"DataC3" = third data item in the sample data from sensor

"Pressure" = fourth data item in the sample data from sensor

"Temperature" = fifth data item in the sample data from sensor

"CompressedData" = all five data items packed into an array of 14 bytes as follows:  
first, second, third, and fourth data items represented as 3 bytes each where the first two bytes are the mantissa value and the third byte is the decimal value \* 100, followed by the fifth data item \* 100 represented as a two byte value.

The PayloadInterface-telemetry-config.xml entry for Turner C3 is:

```
<Element>
  <first boardID="0x20" taskID="0x17" />
  <second>
    <key boardID="0x20" taskID="0x17" />
    <payloadType value="0x00000060" />
    <eventNameToFormatCode>
      <Element>
        <first value="C3SampleHeader" />
        <second value="0" />
      </Element>
      <Element>
        <first value="C3SampleData" />
        <second value="1" />
      </Element>
    </eventNameToFormatCode>
    <reportParams>
      <Element>
        <first value="0" />
        <second maxSamples="1" targetAddress="2">
          <reportHeaderFormat>
            <fields>
              <Element libField="Header91" />
              <Element libField="Raw" eventField="Telem-LengthToFollow" />
              <Element libField="Raw" eventField="Telem-PayloadType" />
              <Element libField="Raw" eventField="Telem-BoardID" />
              <Element libField="Raw" eventField="Telem-TaskID" />
              <Element libField="FC0ASCIINormal" />
              <Element libField="RawINT8" eventField="Telem-NumSamples" />
            </fields>
          </reportHeaderFormat>
          <sampleHeaderFormat>
            <fields>
              <Element libField="Raw" eventField="Telem-Latitude" />
              <Element libField="Raw" eventField="Telem-Longitude" />
              <Element libField="Raw" eventField="Telem-TimeSec" />
              <Element libField="Raw" eventField="C3Event_Header" />
            </fields>
          </sampleHeaderFormat>
        </second>
      </Element>
    </reportParams>
    <Element>
      <first value="1" />
      <second maxSamples="7" targetAddress="2">
        <reportHeaderFormat>
          <fields>
            <Element libField="Header91" />
            <Element libField="Raw" eventField="Telem-LengthToFollow" />
            <Element libField="Raw" eventField="Telem-PayloadType" />
            <Element libField="Raw" eventField="Telem-BoardID" />
            <Element libField="Raw" eventField="Telem-TaskID" />
            <Element libField="FC0BinaryFormat9" />
            <Element libField="RawINT8" eventField="Telem-NumSamples" />
          </fields>
        </reportHeaderFormat>
        <sampleHeaderFormat>
          <fields>
            <Element libField="Raw" eventField="Telem-Latitude" />
            <Element libField="Raw" eventField="Telem-Longitude" />
            <Element libField="Raw" eventField="Telem-TimeSec" />
            <Element libField="Raw" eventField="C3Event_CompressedData" />
          </fields>
        </sampleHeaderFormat>
      </second>
    </Element>
  </second>
</Element>
```

```
        </sampleHeaderFormat>
    </second>
</Element>
</reportParams>
</second>
</Element>
```

### Common XML file keys to edit

**<serialPortSettings>** for C3, the default serial port is /dev/ttyLX4.

By default it is set to communicate with a baudrate of 9600, parity NONE, 1 stopbit, characterSize 8, and flowControl NONE. Ensure that the serial connector cables are plugged into the specified serial port on the SMC.

**<powerOnVerb>** and **<powerOffVerb>** for C3, the command string references power switch. Ensure that the power cable is plugged into the specified power switch on the SMC.

**<defaultOutfile>** for storing all header and data strings received from the C3, on the SMC filesystem. The default specified is c3data.txt in the local directory (/home/smc/roots/200\_C3)

C3 is a command and event interface to a Turner C3 fluorometer; this is a re-implementation of the PIB-based C3 interface, and the commands are kept similar for continuity.

## ConnectionManager Configuration

The ConnectionManager module manages various communication transports, such as a Cell Modem. It listens for requests for a channel (sorted by protocolRevision). It powers up a channel when the first request for a channel is received, and it powers down a channel when the last request is canceled. Additionally, ConnectionManager can fire events with the reported signal strength of a channel.

For Connection Manager to function correctly, the appropriate communication transports (eg, Cell Modem) must be wired up to the Ethernet port and to power switches appropriately.

The ConnectionManager application is started from

**~/roots/150\_ConnectionManager/**

And is configured by the XML file:

**/roots/150\_ConnectionManager/ConnectionManager-config.xml**

ConnectionManager is assigned the board ID 32 (0x20) and the task ID 169 (0xA9).

### Common XML file keys to edit

A sample configuration for the controller settings is as follows:

```
<channelSettings class_id="2" tracking_level="0" version="0">
    <count>2</count>
    <item_version>0</item_version>
    <item>
        <protocolRevision>32</protocolRevision>
        <powerOnVerb>./WiFiPowerOn</powerOnVerb>
        <powerOffVerb>./WiFiPowerOff</powerOffVerb>
        <signalStrengthVerb></signalStrengthVerb>
```

```
        <signalStrengthFilename></signalStrengthFilename>
    </item>
    <item>
        <protocolRevision>33</protocolRevision>
        <powerOnVerb>./CellPowerOn</powerOnVerb>
        <powerOffVerb>./CellPowerOff</powerOffVerb>
        <signalStrengthVerb>./GetSignalStrength_DigiConnect</signalStrengthVerb>
        <signalStrengthFilename>SignalStrength.txt</signalStrengthFilename>
    </item>
</channelSettings>
```

- **count:** This is the number of channels that the ConnectionManager will manage.
- **protocolRevision:** The protocolRevision of the channel. This needs to be unique for everything channel. This is how other processes identify which channel they are requesting.
- **powerOnVerb:** This is the command or script that is executed to turn on the channel.
- **powerOffVerb:** This is the command or script that is executed to turn off the channel.
- **signalStrengthVerb:** This is the command or script executed to retrieve the signal strength of the channel. If left blank, no signal strength will be reported.
- **signalStrengthFilename:** If the signalStrengthVerb is specified, this is the file where the signal strength of the channel is stored.

## DataRecorder Configuration

The DataRecorder application is a module capable of archiving data received, over a serial port, in a data file on the SMC filesystem and making it available for extraction at runtime. The DataRecorder application also provides an interface for switching a pre-specified power switch (that the data sending device may be connected to), changing the communications settings of the serial port, changing the archive data file, and pause-resume recording of the data being received over the serial port. There are two instances of the DataRecorder installed on the SMC, designed to be used independently using separate serial ports and power switches. These are setup and configured in:

~/roots/200\_DataRecorder1 and ~/roots/200\_DataRecorder2

DataRecorder1 instance is configured with the serverName **DataRecorder1**, and assigned the boardId 32 (hexadecimal 20) and taskId 48 (hexadecimal 30).

DataRecorder2 instance is configured with the serverName **DataRecorder2**, and assigned the boardId 32 (hexadecimal 20) and taskId 49 (hexadecimal 31)..

### Common XML file keys to edit

- **serialPortSettings** for DataRecorder1, the default serial port is /dev/ttyLX6 and for DataRecorder2 the default serial port is /dev/ttyLX7. By default, both are set to communicate with a baudrate of 9600, parity NONE, 1 stopbit,

characterSize 8, and flowControl NONE. Ensure that the serial connector cables are plugged into the specified serial ports on the SMC.

- **outFileName** for DataRecorder1, the default archive file to record data received from the serial port is, **recorder1.out** in the default directory ~/roots/200\_DataRecorder1. For DataRecorder2, the default archive file to record data received from the serial port is, **recorder2.out** in the default directory ~/roots/200\_DataRecorder2.
- **powerOnVerb** and **powerOffVerb** for DataRecorder1, the command string references power switch 3 and for DataRecorder2 the command string references power switch 4. Ensure that the power cables are plugged into the specified power switches on the SMC.

### Using the DataRecorder

The DataRecorder application supports the following commands (sent on the SMAC via ServerExec using the serverName of the target instance, or from shore shade applications by specifying the boardId and taskId of the target instance):

#### START

setup recorder in resume (recording) mode and turn on the power switch.

#### STOP

turn off the power and set the recording mode to pause.

#### PAUSE

pause recording of data being received from the serial port (read & discard)

#### RESUME

continue recording data received from the serial port into the active output file.

#### OUTPUT <filename>

record subsequent data to the file <filename>.

#### ON

set power switch to ON

#### OFF

set power switch to OFF

#### SETPORT <baudrate> [ <parity> [ <characterSize> [ <stopBits> ] ] ]

change serial port's settings.

The above commands themselves are not case sensitive, however arguments such as <filename> are used as case-sensitive names.

Example:      `Output /mnt/XDATA/recorder1data.out`  
                 `OUTPUT /mnt/XDATA/Archive1.TXT`  
                 `output /home/smc/roots/200_DataRecorder2/rec2.bin`

## GPSListener Configuration

GPSListener is a service that listens to the NMEA sentences being received from the navigation computer, typically on /dev/ttyS1 or /dev/ttyO1 and:

- Parses the sentences to formulate GPS events
- Transmits GPS events over CORBA
- Sets the system time in based on the content of the event, using the PPS (pulse per second) input to synchronize the timing

### Commands

GPSListener supports the following commands:

#### **GPS AUTOSTART [<state>]**

If the *state* variable is provided, the service is configured to automatically begin storing data to the output file on startup ("true") or is left waiting for a START command ("false")

If the *state* variable is not provided, the current state of the AUTOSTART feature is returned from the command.

#### **GPS DEFAULTOUTPUT [<filename>]**

If the *filename* variable is provided, the default output file name is set to the given value; at startup, the output file will be set to this default file name.

If the *filename* variable is not provided, the default output file is returned from the command.

**GPS OUTPUT [<filename>]** If the *filename* variable is provided, the current output file name is set to the given value.

If the *filename* variable is not provided, the current output file name is returned from the command.

#### **GPS SIMULATE OFF**

Turns off GPS simulation.

**GPS SIMULATE ON <center-latitude> <center-longitude> <speed-kt>  
<radius-m> <CW|CCW>**

Starts GPS point generation using the given variables, assuming that the vehicle is traveling in a circle.

**GPS START**

Commands the service to begin storing received data into the output file. Note that if the AUTOSTART feature is enabled, storage will be automatically started at system startup.

**GPS STOP**

Commands the service to stop storing received data into the output file.

**CORBA Events**

GPSListener sends out GPS CORBA events according to the parameters in the XML file. The maximum receive rate for GPS data is 1Hz, but the XML file can specify events to be sent out at different rates by dividing the seconds down. The typical configuration is as follows:

```
<gpsEventSpecs class_id="3" tracking_level="0" version="0">
  <count>2</count>
  <item_version>1</item_version>
  <item class_id="4" tracking_level="0" version="1">
    <divisor>5</divisor>
    <divisorMode>0</divisorMode>
    <eventSettings class_id="5" tracking_level="0" version="0">
      <eventName>GPSSample</eventName>
      <eventType class_id="6" tracking_level="0" version="0">
        <typeName>GPSSample</typeName>
        <domainName>LiquidR-SMC</domainName>
      </eventType>
    </eventSettings>
  </item>
  <item>
    <divisor>1</divisor>
    <divisorMode>0</divisorMode>
    <eventSettings>
      <eventName>GPSSample1Sec</eventName>
      <eventType>
        <typeName>GPSSample</typeName>
        <domainName>LiquidR-SMC</domainName>
      </eventType>
    </eventSettings>
  </item>
</gpsEventSpecs>
```

In this configuration, there are two events sent out:

- GPSSample is sent out once every 5 seconds
- GPSSample1Sec is sent out once every 1 second

Both are of the same type (typename "GPSSample" and domain name "LiquidR-SMC") By convention the type name is used to indicate which class parses the CORBA data, so EventLib's GPSSample class models CORBA property names:

| Field Name          | Type         | Description                                     |
|---------------------|--------------|-------------------------------------------------|
| GPS-DegreesMagnetic | DOUBLE FLOAT | Course direction in degrees magnetic (from VTG) |

| Field Name              | Type         | Description                                                                                                 |
|-------------------------|--------------|-------------------------------------------------------------------------------------------------------------|
| GPS-DegreesTrue         | DOUBLE FLOAT | Course direction in degrees true (from VTG)                                                                 |
| GPS-Elevation           | DOUBLE FLOAT | Antenna altitude (meters, from GGA)                                                                         |
| GPS-Latitude            | DOUBLE FLOAT | Position/latitude in degrees (from GGA)                                                                     |
| GPS-Longitude           | DOUBLE FLOAT | Position/longitude in degrees (from GGA)                                                                    |
| GPS-MagneticDeclination | DOUBLE FLOAT | Magnetic declinations (from RMC)                                                                            |
| GPS-NMEASentences       | STRING ARRAY | The collection of NMEA sentences comprised in this sample                                                   |
| GPS-NumSatellites       | UINT32       | The number of satellites seen (from GGA)                                                                    |
| GPS-RMSAccuracy         | DOUBLE FLOAT | RMS (root mean square) of the standard deviations of latitude, longitude and altitude (components from RMC) |
| GPS-Simulated           | BOOLEAN      | True if this input was simulated, False if it came from an operating GPS receiver                           |
| GPS-SpeedKmh            | DOUBLE FLOAT | Speed through course in km/h (from VTG)                                                                     |
| GPS-SpeedKnots          | DOUBLE FLOAT | Speed through course in knots (from VTG)                                                                    |
| GPS-TimeSec             | UINT32       | UTC time in seconds since 1-Jan 1970 (from ZDA)                                                             |
| GPS-TimeUsec            | UINT32       | Microseconds component of time (from ZDA)                                                                   |

### XML Configuration

The configuration file (GPSListener-config.xml), which must be located in the startup directory (typically /home/smc/roots/200\_GPSListener), conforms to the format of other configuration files, but here are a few areas of interest:

```
<serialPortSettings class_id="2" tracking_level="0" version="0">
  <portName>/dev/ttyO1</portName>
  <baudRate>4800</baudRate>
  <parity>NONE</parity>
  <characterSize>8</characterSize>
  <stopBits>1</stopBits>
  <flowControl>NONE</flowControl>
  <rts>1</rts>
  <dtr>1</dtr>
  <breakTimeoutMultiplier>1</breakTimeoutMultiplier>
</serialPortSettings>
<maxString>1000</maxString>
<readTimeoutMs>50</readTimeoutMs>
```

Note that the serial port and the baud rate are configurable. *maxString* is the size of the read to be made: be sure that the entire set of strings can fit in that number of characters. *readTimeoutMs* is the inter-byte gap in milliseconds that triggers the data to be returned.

```
<dateSetDirectly>1</dateSetDirectly>
<dateVerb>sudo date --utc {mm}{dd}{HH}{MM}{yyyy}.{SS}</dateVerb>
<hwClockVerb>sudo hwclock -w</hwClockVerb>
<defaultOutputFileName>/mnt/XDATA/GPS/sensor.out</defaultOutputFileName>
```

*dateSetDirectly* should be non-zero if using the Linux system API's to set the clock; that is the most efficient way. If not setting the clock using the system API's, the *dateVerb* can be used to set the time using **date**, but this will result in additional jitter.

Also note that *hwClockVerb* is provided to copy transfer the system date/time to the hardware clock. An accurate hardware clock is useful to set the system time before GPS is received.

```
<deltaToSetTimeMs>100</deltaToSetTimeMs>
<deltaToSetHWClockMs>0</deltaToSetHWClockMs>
```

The system time is not updated unless there is a difference of at least *deltaToSetTimeMs* milliseconds; if it is set, the hardware clock is update if it has not been set before or if at least *deltaToSetHWClockMs* milliseconds has elapsed since the last setting. Note that if *deltaToSetHWClockMs* is zero, the hardware clock is only ever set once.

```
<rmsAccuracyThreshold>50</rmsAccuracyThreshold>
```

If the RMS calculation considering the standard deviation of latitude, longitude and altitude exceeds *rmsAccuracyThreshold*, the GPS sample is not sent.

```
<gpsEventSpecs class_id="3" tracking_level="0" version="0">
  <count>2</count>
  <item_version>1</item_version>
  <item class_id="4" tracking_level="0" version="1">
    <divisor>5</divisor>
    <divisorMode>0</divisorMode>
    <eventSettings class_id="5" tracking_level="0" version="0">
      <eventName>GPSSample</eventName>
      <eventType class_id="6" tracking_level="0" version="0">
        <typeName>GPSSample</typeName>
        <domainName>LiquidR-SMC</domainName>
      </eventType>
    </eventSettings>
  </item>
  <item>
    <divisor>1</divisor>
    <divisorMode>0</divisorMode>
    <eventSettings>
      <eventName>GPSSample1Sec</eventName>
      <eventType>
        <typeName>GPSSample</typeName>
        <domainName>LiquidR-SMC</domainName>
      </eventType>
    </eventSettings>
  </item>
</gpsEventSpecs>
```

This section configures how many events are sent out and on what frequency. In this case, there are two events being sent:

- "GPSSample" reports GPS on 5-second boundaries
- "GPSSample1Sec" reports GPS on 1-second boundaries

The number of events can be customized to reduce overhead or increase resolution.

```
<tryBaudRates>
  <count>2</count>
  <item_version>0</item_version>
  <item>4800</item>
  <item>9600</item>
</tryBaudRates>
```

This section determines which baud rates will be tried: SV2 is set up to report GPS at 4800 baud, while SV3 reports it at 9600 baud.

```
<setTimeOnPPS>true</setTimeOnPPS>
<ppsGPIO>167</ppsGPIO>
```

If *setTimeOnPPS* is true, the system date/time will only be updated on the edge of the PPS (pulse per second) output of the GPS receiver. The signal being monitored is configured by *ppsGPIO*

## Logger

The Logger utility can be used in place of **tee** to write log files from process output, and it allows limitations to be placed on the size of log files produced. The following help is available by issuing **Logger --help**:

```
Logger: Logger { ... options ... }

Options can be specified as a collection of single-letter shortcuts
preceded by a dash and followed by tokens to use as arguments or as long-
form options preceded by double-dashes with arguments formatted as
assignments.  For example:

    Logger -wc /my-partition/workfiles BZIP2 --prefix=MYLOG --
    extension=.EXT --max-files 10 --max-file-size 1000000

The 'w' and 'c' short form options are assigned '/my-partition/workfiles'
and 'BZIP2'; the --prefix option is assigned 'MYLOG'; the --extension
option is assigned '.EXT'; the --max-files option is assigned '10'; and
the --max-file-size option is assigned '1000000'

The available options are as follows:
```

|                            |                                                                                                    |
|----------------------------|----------------------------------------------------------------------------------------------------|
| -a, --archive-dir          | The directory where log archives are stored (defaults to work directory)                           |
| -c, --archive-compression  | The type of archive compression to use: BZIP2, GZIP or NONE                                        |
| -e, --extension            | The extension used for each output file (before compression)                                       |
| -f, --max-files            | The maximum number of files (archives+work file) to retain                                         |
| -g, --purge-strategy       | The strategy to use when purging files: OLDEST or NONE                                             |
| -k, --keep-writes-together | Allow lines to break across files (0) or keep writes/lines together (1)                            |
| -p, --prefix               | The first part of the file name for each file in the set                                           |
| -q, --max-bytes-queued     | The size of the inter-thread write queue in bytes (larger value delays the logging client less)    |
| -r, --max-retain-time      | The maximum number of seconds to retain archives (0=no limit)                                      |
| -s, --max-file-size        | The maximum size of each file (before compression)                                                 |
| -w, --work-dir             | The directory for the work file and default for archive files; starting directory if not specified |

## GPSWaves Configuration (GPS and IMU based waves sensor )

Interface Microstrain 3DM-GX35 GPS and IMU sensor and use its GPS output to compute wave height parameters (using Jim Thomson's algorithms.) Keep the software that Jim Thomson produces in a separate library so that it can be substituted separately from the main program.

## Commands

Commands to GPSWaves are sent to the process name "GPSWaves" with ServerExec or it is typically assigned an address of 0x2025 (the upper byte varies with the base address of the SMC.) The available commands are:

### **GWAVES --autostart [<new-state>]**

If the *new-state* parameter is given, it sets the state of the automatic start feature. If *new-state* is **true**, the service starts sampling and returning data when the program launches; otherwise, the program remains idle after launch and an explicit start command must be sent to start sampling.

If the *new-state* parameter is omitted, the command returns the current state of the automatic start feature: **true** (begin sampling on program launch) or **false** (idle on program launch.)

### **GWAVES --period <minutes>**

Specifies the period, in minutes, over which the Microstrain data samples are collected for processing by Jim Thomson's library, for calculating wave height and spectrum. **<minutes>** can be any value between 0 and 60, but values lower than 20 minutes are not recommended as that will result in insufficient data samples to perform wave height calculations. The system default is set at 30 minutes.

### **GWAVES --start**

Powers on the Microstrain IMU and begins sampling, storing, and processing its data for wave height calculations.

### **GWAVES --stop**

Stops sampling and sending wave height calculations and powers down the Microstrain.

## CORBA Events

GPSWaves sends out one event (GPSWavesTelemEvent) to report the wave height calculations for each period (duration of which is specified as described above); it contains the following fields:

| Field Name | Type         | Description                                                |
|------------|--------------|------------------------------------------------------------|
| Hs         | DOUBLE FLOAT | Wave height (in meters)                                    |
| Tp         | DOUBLE FLOAT | Peak period (in seconds)                                   |
| Dp         | DOUBLE FLOAT | Dominant direction ( in degrees)                           |
| Samples    | UINT16       | Number of samples used for this calculation                |
| Fs         | DOUBLE FLOAT | Frequency of data samples during this period's calculation |
| SampleGaps | UINT16       | Number of missing/dropped samples during this period       |

The event is sent back to WGMS using the following entry in PayloadInterface-telemetry-config.xml:

```
<Element>
  <first boardID="0xFF" taskID="0x25" />
  <second>
    <key boardID="0xFF" taskID="0x25" />
    <payloadType value="0x00000093" />
  </second>
</Element>
```

```

        <eventNameToFormatCode>
        <Element>
            <first value="GPSWavesTelem" />
            <second value="0" />
        </Element>
    </eventNameToFormatCode>
    <reportParams>
    <Element>
        <first value="0" />
        <second maxSamples="1" targetAddress="2">
            <reportHeaderFormat>
            <fields>
                <Element libField="Header91" />
                <Element libField="Raw" eventField="Telem-
LengthToFollow" />
                <Element libField="Raw" eventField="Telem-PayloadType"
/>
                <Element libField="Raw" eventField="Telem-BoardID" />
                <Element libField="Raw" eventField="Telem-TaskID" />
                <Element libField="FC0BinaryNormal" />
                <Element libField="RawINT8" eventField="Telem-
NumSamples" />
            </fields>
            </reportHeaderFormat>
            <sampleHeaderFormat>
            <fields>
                <Element libField="Raw" eventField="Telem-Latitude" />
                <Element libField="Raw" eventField="Telem-Longitude" />
                <Element libField="Raw" eventField="Telem-TimeSec" />
                <Element libField="Raw"
eventField="GPSWavesTelemEvent_Hs" />
                <Element libField="Raw"
eventField="GPSWavesTelemEvent_Tp" />
                <Element libField="Raw"
eventField="GPSWavesTelemEvent_Dp" />
                <Element libField="Raw"
eventField="GPSWavesTelemEvent_Samples" />
                <Element libField="Raw"
eventField="GPSWavesTelemEvent_Fs" />
                <Element libField="Raw"
eventField="GPSWavesTelemEvent_SampleGaps" />
            </fields>
            </sampleHeaderFormat>
        </second>
    </Element>
</reportParams>
</second>
</Element>

```

### XML Configuration

The configuration file (GPSWaves-config.xml), which must be located in the startup directory (typically /home/smc/roots/200\_GPSWaves), conforms to the format of other configuration files, but the following items are of particular interest:

```

<controllerSettings class_id="12" tracking_level="0" version="1">
    <serialPortSettings class_id="13" tracking_level="0" version="0">
        <portName>/dev/ttyLX4</portName>
        <baudRate>115200</baudRate>
        <parity>NONE</parity>
        <characterSize>8</characterSize>
        <stopBits>2</stopBits>
        <flowControl>NONE</flowControl>
        <rts>1</rts>
    </serialPortSettings>
</controllerSettings>

```

```

        <dtr>1</dtr>

        <breakTimeoutMultiplier>1</breakTimeoutMultiplier>
    </serialPortSettings>
    <powerOnVerb>sudo SetPowerSwitch 2 1</powerOnVerb>
    <powerOffVerb>sudo SetPowerSwitch 2 0</powerOffVerb>
    <outputFileName>/mnt/XDATA/GPSWaves/rawdata.txt</outputFileName>

    <processedDataFileName>/mnt/XDATA/GPSWaves/GPSwaves.txt</processedDataFile
Name>

    <spectrumFileName>/mnt/XDATA/GPSWaves/spectrum.txt</spectrumFileName>
    <samplingFrequency1000>4000</samplingFrequency1000>
    <seaStateInterval>30</seaStateInterval>
    <lfnr>2</lfnr>
    <nStdDev>5</nStdDev>
    <enableRawDataEvent>false</enableRawDataEvent>
    <enableSpectrumEvent>false</enableSpectrumEvent>
    <enableTelemEvent>true</enableTelemEvent>
</controllerSettings>

```

The **samplingFrequency1000** must match the Microstrain's GPS data frequency configuration and is the frequency\*1000

## LuaWrapper Configuration

### Commands

The following commands can be sent to LuaWrapper via ServerExec or by targeting its address with Write Console Script:

LWM -list

Lists the lua instance names and the file that was loaded into each instance.

LWM --loadlua *lua-script-file instance-name*

Creates a new lua instance named for *instance-name* and initializes with the script file indicated by *lua-script-file*

LWM --unloadlua *instance-name*

Unloads the lua instance previously-loaded as *instance-name*

LWM --run *command-file*

Loads the text lines from *command-file* and executes them as a set of LuaWrapper commands.

LWSCRIPT --invoke *instance-name* *function-name* { ...*argument-list*... }

Invokes the function named *function-name* on the script that is running in the previously-loaded instance, *instance-name*

### Special Variables

LuaWrapper makes the following variables and meta-tables available to the script:

- SMC, which exposes the primary API to the script
- SMCContext, which exposes an invocation-specific API to the script to control when the **LWSCRIPT --invoke** command returns
- A serial port meta-table that is returned by SMC:NewSerialPort
- A timer meta-table that is returned by SMC:NewTimer

The SMC meta-table is as follows:

*v1 bitwise-and v2* = SMC:BAnd(*v1*,*v2*)

Converts *v1* and *v2* to unsigned 32-bit integers and then performs a bitwise AND, returning the result. The truth table for each bit is as follows:

| v1 bit | v2 bit | result bit |
|--------|--------|------------|
| 0      | 0      | 0          |
| 1      | 0      | 0          |
| 0      | 1      | 0          |
| 1      | 1      | 1          |

*v1 bitwise-and v2* = SMC:BOr(*v1*,*v2*)

Converts *v1* and *v2* to unsigned 32-bit integers and then performs a bitwise OR, returning the result. The truth table for each bit is as follows:

| v1 bit | v2 bit | result bit |
|--------|--------|------------|
| 0      | 0      | 0          |
| 1      | 0      | 1          |
| 0      | 1      | 1          |
| 1      | 1      | 1          |

*v1 bitwise-and v2* = SMC:BXor(*v1*,*v2*)

Converts *v1* and *v2* to unsigned 32-bit integers and then performs a bitwise XOR, returning the result. The truth table for each bit is as follows:

| v1 bit | v2 bit | result bit |
|--------|--------|------------|
| 0      | 0      | 0          |
| 1      | 0      | 1          |
| 0      | 1      | 1          |
| 1      | 1      | 0          |

*gps-reading* = SMC:GetLocation()

Returns the last GPS reading as a table with elements "latitude" and "longitude" set to the whole and fractional degrees; if no GPS reading has been received, GetLocation returns nil. For example:

```
local lastReading = SMC:GetLocation()
if lastReading then
    print("Last GPS reading: ("
        ..tostring(lastReading.latitude)..", "
        ..tostring(lastReading.longitude)..")")
end
```

SMC:Log(*log-level*, *message*)

Outputs the *message* string to the process log at a priority (*log-level*) which may be any of: "debug", "info", "notice", "warning", "error", "critical", "alert" or "emergency"

*timsp-var* = SMC:NewSerialPort(*config-ID*)

Creates an instance of a serial port object based on the selected configuration (*config-ID*, a string.) The returned data type is a serial port meta-table (see below) Received data is posted to the lua script via the **SMCReportSerialData** function, if it is defined.

*timer-var* = SMC:NewTimer(*timer-ID*, *duration*)

Creates an instance of a timer object with the given ID (*timer-ID*, a string) and *duration* (a number of milliseconds.) If the timer is not deleted before the timeout expires, the script's **SMCReportTimeout** function is invoked, if it is defined.

SMC:PublishDataSet(*event-name*, *event-type*, *event-domain*, *data-set*, *content*)

Publishes a data set event with event characteristics set by *event-name*, *event-type* and *event-domain* (all strings.) In addition, the data set name (*data-set*, a string) and the *content* of the data set (a string) are provided.

| <b><i>event-name</i></b> | <b><i>event-type</i></b> | <b><i>event-domain</i></b> | <b><i>destination</i></b> |
|--------------------------|--------------------------|----------------------------|---------------------------|
| PPPHostNewFileAvailable  | DataSetEvent             | LiquidR-SMC                | WGMS via Iridium SBD      |
| PyldNewFileAvailable     | DataSetEvent             | LiquidR-SMC                | WGMS via Iridium SBD      |

SMC:PublishEvent(*event-table*)

Uses *event-table* to create a CORBA event and publish it to EventManager. See the CORBA Event Table section, below.

`SMC:SendCommand(server-name,variable,invoke-synchronously)`

Performs an invocation via CORBA ICommandable interface on the server named “*server-name*” (must be a string), passing the “*variable*” (string, number, table of strings or table of numbers) as the content.

`uptime-seconds = SMC:SystemUptime()`

Returns the number of seconds since the SMC was started as a floating point number with a resolution of 0.01 seconds.

The SMCContext meta-table is as follows:

`SMCContext:AppendOutput(data)`

Appends *data* string to the output that will be reported from the command.

`SMCContext:ReportComplete()`

For a command that was previously reported as asynchronous (SMCContext:SetAsynch), this indicates that the command is now complete.

`SMCContext:SetAsynch()`

Indicates that the invocation (LWSCRIPT --invoke) corresponding to this context variable should not return until ReportComplete() is called. This allows a more complex state machine to be implemented as a result of the lua script function being called.

`SMCContext:SetSuccessFlag(status)`

Sets or clears the success flag according the boolean variable “*status*” This value is carried back in the command completion event for use by the caller. Note that if an exception is thrown, command is considered to be unsuccessful; if SetSuccessFlag is not called and no exception is thrown, the command is considered to be successful.

The serial port meta-table (returned from SMC:NewSerialPort) is as follows:

`ID = sp-var:GetID()`

Returns the serial port configuration ID string associated with this serial port.

`sp-var:SetBreak()`

Puts the serial port in break mode (continually transmitting electrical low) for the default period of time.

`sp-var:SetBreakTime(ms)`

Puts the serial port in break mode (continually transmitting electrical low) for the specified number of milliseconds (*ms*.)

*sp-var*:Transmit(*data*)

Transmits the *data* string over the serial port.

The timer meta-table (returned from SMC:NewTimer) is as follows:

*ID* = *timer-var*:GetID()

Returns the timer ID associated with the timer.

### CORBA Event Tables and Variable Definitions

CORBA events are used to report significant occurrences and their data to other interested processes via the EventManager interface. Each event has:

- A **name**, which indicates what has occurred.
- A **type**, consisting of a **type name** and a **domain name**; these help to categorize the event report. By convention, the SMC software uses the event **type** to define how the accompanying data is formatted.
- A series of **name/value** pairs that carry data relative to the event report.

Each of the fields above is a string with the exception of the **value**; each value can take on one of the following CORBA types:

| CORBA Type ID       | Meaning                                  | Example(s) in lua                                              |
|---------------------|------------------------------------------|----------------------------------------------------------------|
| BOOLEAN             | Boolean value or flag                    | true or false                                                  |
| BOOLEAN_VECTOR      | array of Boolean values or flags         | { { true, false, false, true } }                               |
| DOUBLE_FLOAT        | double-precision floating point          | <i>number</i> or 3.141592654                                   |
| DOUBLE_FLOAT_VECTOR | array of double-precision floating point | { { 1, 2, 3, 4 } }                                             |
| INT16               | 16-bit signed integer                    | { <i>number</i> , "INT16" } or { 8193, "INT16" }               |
| INT16_VECTOR        | array of 16-bit signed integers          | { { 1, 2, 3, 4 }, "INT16_VECTOR" }                             |
| INT32               | 32-bit signed integer                    | { <i>number</i> , "INT32" } or { -1000000, "INT32" }           |
| INT32_VECTOR        | array of 32-bit signed integers          | { { 1, 2, 3, 4 }, "INT32_VECTOR" }                             |
| INT8                | 8-bit signed integer                     | { <i>number</i> , "INT8" } or { 5, "INT8" } or { -5, "INT8" }  |
| INT8_VECTOR         | array of 8-bit signed integers           | { { 1, 2, 3, 4 }, "INT8_VECTOR" }                              |
| NULL                | nil                                      | Nil                                                            |
| SINGLE_FLOAT        | single-precision floating point          | { <i>number</i> , "SINGLE_FLOAT" } or { 3.14, "SINGLE_FLOAT" } |
| SINGLE_FLOAT_VECTOR | array of single-precision floating point | { { 1, 2, 3, 4 }, "SINGLE_VECTOR" }                            |
| STRING              | character string                         | "Hello, world" or { 12, "STRING" }                             |
| STRING_VECTOR       | array of character strings               | { { "Mercury", "Venus", "Earth", "Mars" } }                    |
| UINT16              | 16-bit unsigned integer                  | { <i>number</i> , "UINT16" } or { 40000, "UINT16" }            |
| UINT16_VECTOR       | array of 16-bit unsigned integers        | { { 1, 2, 3, 4 }, "UINT16_VECTOR" }                            |
| UINT32              | 32-bit unsigned integer                  | { <i>number</i> , "UINT32" } or { 1000000, "UINT32" }          |
| UINT32_VECTOR       | array of 32-bit unsigned integers        | { { 1, 2, 3, 4 }, "UINT32_VECTOR" }                            |
| UINT8               | 8-bit unsigned integer                   | { <i>number</i> , "UINT8" } or { 12, "UINT8" }                 |
| UINT8_VECTOR        | array of 8-bit unsigned integers         | { { 1, 2, 3, 4 }, "UINT8_VECTOR" }                             |

The generic form of a CORBA event in lua table format is:

```
{ header =      { name = "event-name",
                  typename = "event-type-name",
                  domainname = "event-domain-name" },
  properties = { prop1name = prop1value,
                  prop2name = prop2value,
                  ...
                  propnname = propnvalue } }
```

For example, to publish an event:

```
local evt = { }

evt.header = { name = "LWTelem1",
               typename = "LWTelem1",
               domainname="LqrSMAC" }

evt.properties = { }
evt.properties["Telem-BoardID"] = { 32, "UINT8" }
evt.properties["Telem-TaskID"] = { 64, "UINT8" }
evt.properties["StringMessage"] = "Hello, world!"

SMC:PublishEvent(evt)
```

To receive an event and print out the fields to the console:

```
function SMCReportEvent(eventRx)

    print("SMCReportEvent name="..tostring(eventRx.header.name))
    print("header " ..tostring(eventRx.header))
    print("header.name " ..tostring(eventRx.header.name))
    print("header.typename " ..tostring(eventRx.header.typename))
    print("header.domainname " ..tostring(eventRx.header.domainname))

    print("properties " ..tostring(eventRx.properties))

    for k,v in pairs(eventRx.properties)
    do
        if type(v[1]) == "table" then
            local s2 = ""
            for k2,v2 in ipairs(v[1]) do
                if #s2 ~= 0 then
                    s2 = s2..","
                end
                s2 = s2..tostring(k2)..":"..tostring(v2)
            end
            print("properties["..k.."] ->
"..type(v[1]).."/"..s2.." " ..tostring(v[2]))
        else
            print("properties["..k.."] ->
"..type(v[1]).."/"..tostring(v[1]).." " ..tostring(v[2]))
        end
    end
end
```

Here's a simplified way of handling the typical GPS distribution event as a generic event:

```
function SMCReportEvent(eventRx)

    if tostring(eventRx.header.name) == "GPSSample" then

        local latitude = eventRx.properties["GPS-Latitude"][1]
        local longitude = eventRx.properties["GPS-Longitude"][1]

        if latitude and longitude then
            print("gps latitude="..tostring(latitude)..
                longitude="..tostring(longitude))
            end
        end
    end

end
```

Sometimes string data will be sent out as an array of bytes (UINT8\_VECTOR.)  
Byte arrays can be converted to string as follows:

```
local field = eventRx.properties["some-property"]
local data = field[1][1]

if type(data)=="table" and tostring(field[2][2]) == "UINT8_VECTOR" then
    data = string.char(unpack(data))
end
```

### Pre-defined Callback Functions

LuaWrapper reserves the following script functions for call-backs:

**function SMCReportEvent(eventRx)**

This function is called whenever an event is received and is successfully converted to a lua table (see CORBA Event Tables and Variable Definitions, above.) Note that if the Event Transceiver settings in the XML have enabled the event filter and the received event is filtered out, SMCReportEvent not called.

**function SMCReportSerialData(ID,data)**

This function is called whenever serial data is received according to the configuration specified in the XML and referenced when opening the port:

**ID** is a string matching the configuration name passed to SMC:NewSerialPort;

**data** is a string containing the received data.

**function SMCReportTimeout(ID)**

This function is called whenever a timer expires:

**ID** is a string matching the timer ID passed to SMC:NewTimer

## XML Configuration

The configuration file (LuaWrapper-config.xml), which must be located in the startup directory (typically /home/smc/roots/200\_LuaWrapper), conforms to the format of other configuration files, but the controllerSettings section is of particular interest:

```
<?xml version="1.0"?>
<controllerSettings class_id="13" tracking_level="0" version="1">
  <serialConfigMap class_id="14" tracking_level="0" version="0">
    <count>1</count>
    <item_version>0</item_version>
    <item class_id="15" tracking_level="0" version="0">
      <first>config1</first>
      <second class_id="16" tracking_level="0" version="1">
        <listenerSettings class_id="17" tracking_level="0" version="0">
          <maxReadSize>4096</maxReadSize>
          <intercharacterGapTimeoutMs>50</intercharacterGapTimeoutMs>
        </listenerSettings>
        <portSettings class_id="18" tracking_level="0" version="0">
          <portName>/dev/ttyLX2</portName>
          <baudRate>9600</baudRate>
          <parity>NONE</parity>
          <characterSize>8</characterSize>
          <stopBits>1</stopBits>
          <flowControl>NONE</flowControl>
          <rts>1</rts>
          <dtr>1</dtr>
          <breakTimeoutMultiplier>1</breakTimeoutMultiplier>
        </portSettings>
      </second>
    </item>
  </serialConfigMap>
</controllerSettings>
```

The **serialConfigMap** is a dictionary of settings (**second** section) for serial ports keyed by configuration name (**first** section); the configuration name is referenced by the SMC:NewSerialPort() function when opening a serial port. Various named configurations can be created in the XML file by repeating the **item** section and varying the configuration name. Note that **count** must be set to the number of **item** sections that will appear within the **serialConfigMap**.

## Sample lua script

The following is a test case that uses some of the features above. Although the scripting interface is necessarily event-oriented to keep it single-threaded, this script shows how lua coroutines can be used to implement a top-down application that sends data to a serial port and listens for responses.

```
-- "Top-down" application module using co-routines to execute
-- sporadically on I/O and timing events

function Application()

    SMC:Log("debug", "Starting Application")

    port = SMC:NewSerialPort("config1")

    local firstTime, thisTime, delta, waitTime

    thisTime = SMC:SystemUptime()
    firstTime = thisTime-1

    while true
    do
        delta = thisTime-firstTime
        waitTime = delta-math.floor(delta)
```

```

        if (waitTime >= 0) and (waitTime < 1) then
            waitTime = math.ceil(1000*(1-waitTime))
        else
            waitTime = 1000
        end

        SMC:Log("debug", "Waiting "..tostring(waitTime))
        Wait(waitTime)
        SMC:Log("debug", "Back from waiting")

        local i
        local awaitResult

        i = 1
        awaitResult = false

        while (i <= 3) and (awaitResult == false)
        do
            port:Transmit("AT\r")
            awaitResult = Await("OK\r\n", 300)
            i = i+1
        end

        thisTime = SMC:SystemUptime()

    end

end

-- Waits for the given serial sequence to be received within the given
-- number of milliseconds; returns true if successful.

function Await(sequence, timeout)

    SMC:Log("debug", "Await sequence="..sequence.." timeout="..timeout)

    local rxd, result

    awaitTimer = SMC:NewTimer("await", timeout)
    result = false

    while (awaitTimer ~= nil) and (result == false)
    do
        rxd = coroutine.yield()
        if (rxd == sequence) then result = true end
    end

    SMC:Log("debug", "Await rxd="..rxd.." at loop exit")

    awaitTimer = nil
    collectgarbage("collect")

    SMC:Log("debug", "Await:"..tostring(result))
    return result

end

-- Delays for the given number of milliseconds

function Wait(timeout)

    waitTimer = SMC:NewTimer("wait", timeout)

    while waitTimer ~= nil
    do
        coroutine.yield()
    end

end

end

```

```

-- Callback routine for serial data received

function SMCReportSerialData(portName,data)

    SMC:Log("debug", "Recieved "..data.." on "..portName)

    if awaitTimer ~= nil then
        awaitTimer = nil
        collectgarbage("collect")
        coroutine.resume(appRtn, data)
    end

end

-- Callback routine for timer expirations

function SMCReportTimeout(timerName)

    SMC:Log("debug", "SMCReportTimeout timerName="..timerName)

    if timerName == "wait" then
        waitTimer = nil
        collectgarbage("collect")
        coroutine.resume(appRtn, nil)
    elseif timerName == "await" then
        awaitTimer = nil
        collectgarbage("collect")
        coroutine.resume(appRtn, nil)
    end

    SMC:Log("debug", "exiting SMCReportTimeout")

end

-- Initialize the module

SMC:Log("debug", "Starting test4")
waitTimer = nil
awaitTimer = nil

-- Launch the application on file load

appRtn = coroutine.create(function()
    Application()
end)
coroutine.resume(appRtn)

SMC:Log("debug", "test4 started")

```

## Magnetometer Configuration (Marine Magnetics Explorer)

Magnetometer interfaces a Marine Magnetics Explorer and collects the data that it streams. A single sample is returned once per minute as telemetry and the interface provides the option to return logged data.

### Commands

Commands to Magnetometer are sent to process name "Magnetometer" with ServerExec or it is typically assigned an address of 0x201D (the upper byte varies with the base address of the SMC.) The available commands are:

**MAG -run <filename>**

Sequentially executes the commands in the file of the given name (*filename*)

MAG --autostart [<new-state>]

If the *new-state* parameter is given, it sets the state of the automatic start feature. If *new-state* is true, the service starts sampling and returning data when the program launches; otherwise, the program remains idle after launch and an explicit start command must be sent to start sampling.

If the *new-state* parameter is omitted, the command returns the current state of the automatic start feature: true (begin sampling on program launch) or false (idle on program launch.)

MAG --getdata [--bzip2] <timestamp>

Retrieves the data received starting at the given timestamp, which must be in `yyyymmddHHMM` form:

yyyy = year  
mm = month  
dd = day of month  
HH = hour  
MM = minute

The entire minute of data is returned, up to 60 lines of text. If the **--bzip2** option is specified, the output is compressed before it is returned.

If compression is used and command is issued from WGMS, please use function code 1 so that the result is returned as a file. Note that function code 0 results to WGMS are limited to only a few messages, so it may be wise to use function code 1 in either case.

MAG --start

Powers up the Explorer and begins sampling and storing data.

MAG --stop

Stops sampling and powers down the Explorer.

MAG --syncclock

Immediately sets the clock on the Explorer to match the SMC's system time. Note that clock is also set when communication to the Explorer is started and at a configurable interval (see XML file description, below.)

MAG --zeropressure

Immediately zero's the pressure sensor reading at the current level; this is useful during deployment of the sensor to take out any offset resulting from shipping.

**CORBA Events**

Magnetometer sends out one event (MagnetometerRawData) to report the sample taken at zero seconds every minute; it just carries the literal string received from the Explorer:

| Field Name         | Type   | Description                                                              |
|--------------------|--------|--------------------------------------------------------------------------|
| StringEvent-String | STRING | The literal message received from the Explorer at offset 0 in the minute |

The event is sent back to WGMS using the following entry in PayloadInterface-telemetry-config.xml:

```
<?xml version="1.0"?>
<Element>
  <first boardID="0xFF" taskID="0x1D"/>
  <second>
    <key boardID="0xFF" taskID="0x1D"/>
    <payloadType value="0x0000008C"/>
    <eventNameToFormatCode>
      <Element>
        <first value="MagnetometerRawData"/>
        <second value="0"/>
      </Element>
    </eventNameToFormatCode>
    <reportParams>
      <Element>
        <first value="0"/>
        <second maxSamples="2" targetAddress="2">
          <reportHeaderFormat>
            <fields>
              <Element libField="Header91"/>
              <Element libField="Raw" eventField="TelemLengthToFollow"/>
              <Element libField="Raw" eventField="TelemPayloadType"/>
              <Element libField="Raw" eventField="TelemBoardID"/>
              <Element libField="Raw" eventField="TelemTaskID"/>
              <Element libField="FC0ASCIIINormal"/>
              <Element libField="RawINT8" eventField="TelemNumSamples"/>
            </fields>
          </reportHeaderFormat>
          <sampleHeaderFormat>
            <fields>
              <Element libField="Raw" eventField="TelemLatitude"/>
              <Element libField="Raw" eventField="TelemLongitude"/>
              <Element libField="Raw" eventField="TelemTimeSec"/>
              <Element libField="Raw" eventField="StringEventString"/>
            </fields>
          </sampleHeaderFormat>
        </second>
      </Element>
    </reportParams>
  </second>
</Element>
```

## XML Configuration

The configuration file (Magnetometer-config.xml), which must be located in the startup directory (typically /home/smc/roots/200\_Magnetometer), conforms to the format of other configuration files, but the following keys are of particular interest:

```
<clockUpdateInterval class_id="14" tracking_level="0" version="0">
  <is_special>0</is_special>
  <time_duration_hours>24</time_duration_hours>
  <time_duration_minutes>0</time_duration_minutes>
  <time_duration_seconds>0</time_duration_seconds>
  <time_duration_fractional_seconds>0</time_duration_fractional_seconds>
</clockUpdateInterval>
```

*clockUpdateInterval* configures the time interval between synchronizing the Explorer's clock to the SMC's system time.

```
<fileSetSettings class_id="18" tracking_level="0" version="0">
  <archiveCompressionType>BZIP2</archiveCompressionType>
  <archiveDirectory></archiveDirectory>
  <extension>.csv</extension>
  <keepWritesTogether>1</keepWritesTogether>
  <maxBytesQueued>10000</maxBytesQueued>
  <maxFileSize>1000000</maxFileSize>
  <maxFiles>1</maxFiles>
  <maxRetainTime>00:00:00</maxRetainTime>
  <prefix>MAGDATA</prefix>
  <purgeStrategy>NONE</purgeStrategy>
  <workingDirectory>/mnt/XDATA/Magnetometer/MAGDATA</workingDirectory>
</fileSetSettings>
```

*fileSetSettings* configures where and how the raw magnetometer readings are stored in the filesystem.

- Files are stored under "/mnt/XDATA/Magnetometer/MAGDATA" (*workingDirectory*)
- Files are named as MAGDATA\*.csv (*prefix* and *extension*) and can be up to 1,000,000 bytes (*maxFileSize*); at that threshold, files are compressed using "bzip2" (*archiveCompressionType*; alternatives are "none" or "gzip")
- There is no enforcement of the maximum number of files because *purgeStrategy* is set to "none"; the alternative is "oldest" which causes the oldest files to be removed when *maxFiles* or *maxRetainTime* is violated.
- There is no separate archive directory, but if one were specified, the renamed and compressed files would be moved there instead of staying in the working directory.

## MAM Configuration (Datawell MOSE)

MAM (MOSE Access Manager) interface a Datawell MOSE motion sensor and implements algorithms for computing wave spectrum and related summary values. This section indicates what has changed and gives some setup guidance, but please consult the LRI Datawell Sensor User's Guide and the MOSE documentation for additional detail.

### Commands

Commands to MAM are sent to process name "MOSESensor" with ServerExec or it is typically assigned an address of 0x2022 (the upper byte varies with the base address of the SMC.) The available commands are:

MOSE --getconfig

Reads the configuration from the sensor by issuing command "\$PDTWL,CFG\*"

MOSE --off

Turns off power to the sensor by executing the power off verb from the XML file.

MOSE --on

Turns on power to the sensor by executing the power on verb from the XML file.

MOSE -o *file-name*

Sets the name and path of the file to which raw samples are logged.

MOSE --output *file-name*

This is a synonym for MOSE -o

1.1.6 MOSE --procout *file-name*

Sets the name and path of the file to which wave summary data is logged; this includes wave height (hs), wave average (tz), dominant wave period (tp) and wave direction (dirp.)

MOSE --reportrate *new-rate*

Configures the number of minutes between spectrum calculations (1..60)

MOSE --reset

Resets the sensor by sending the "\$PDTWL,RST\*" command.

MOSE --setconfig *LFSP LFLP HLFP SA-resist*

Sends configuration parameters to the sensor:

- **LFSP** is the low frequency, short period cut off: 1 = 10s, 2 = 30s
- **LFLP** is the low frequency, long period cut off: 1 = 300s, 2 = 600s, 3 = 1000s
- **HLFP** is high frequency, long period cut off: 1 = 30s, 2 = 60s, 3 = 100s
- **SA-resist**: 0 = off, 1 = ON

All four values may be specified, asterisks may be specified for unchanged parameters, or trailing parameters may be omitted if the values are unchanged.

1.1.6 MOSE --specout *file-name*

Sets the name and path of the file to which Datawell spectrum datasets are logged.

1.1.6 MOSE --spec2out *file-name*

Sets the name and path of the file to which NDBC-compatible wave spectrum data is logged.

MOSE --start

Begins logging data to files as configured.

MOSE --stop

Stops logging data to files.

MOSE --version

Reads the MOSE software version from the sensor by issuing the "\$PDTWL,VER\*" command.

#### Common XML file keys to edit

- **controllerSettings** → **samplesPerSpectrum** is the count of samples to include in a spectrum sample; it is typically set to 256 and must be set to a power of 2.
- **controllerSettings** → **seaStateInterval** is the number of minutes between spectrum calculations; it is typically set to 60, so the spectrum measurement is taken once per hour.
- **controllerSettings** → **includeBadSamples** can be set to a non-zero value for debugging, but should be deployed set to zero (no bad samples are included in calculations.)
- **controllerSettings** → **stabilizedDuration** is the number of minutes to allow for GPS acquisition. MAM may turn the the sensor off to save power, but it must allow some startup time.
- **controllerSettings** → **collectingDuration** is the number of minutes to allow for collection of samples (sampleNum); this time in addition to stabilizedDuration is the minimum "on" time that MAM budgets for a spectrum collection.

### PayloadInterface Configuration

PayloadInterface implements the SV2 Payload protocol or the SV3 REST API in order to communicate with the Float C&C and other devices on the Float and to interact with WGMS via file transfer and telemetry reports. Starting in revision 2.1.0, the user must select which protocol is used with the **sv3Enabled** key:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!DOCTYPE boost_serialization>
<boost_serialization signature="serialization::archive" version="10">
<rootlevel class_id="0" tracking_level="0" version="2">
  <loggerSettings class_id="1" tracking_level="0" version="0">
    <logLevel>LT_INFO</logLevel>
    <processName>PayloadInterface</processName>
    <isFileDes>1</isFileDes>
    <fileDes>1</fileDes>
    <fileName></fileName>
    <pipeName></pipeName>
  </loggerSettings>
  <sv3Enabled>true</sv3Enabled>
  <serialPortSettings class_id="2" tracking_level="0" version="0">
```

The value is set to **true** if PayloadInterface is using the SV3 REST API or **false** if it's using the payload protocol.

PayloadInterface doesn't typically require any configuration, except to ensure that the required telemetry settings are merged into **PayloadInterface-telemetry-**

**config.xml** The shipping version contains the superset of the telemetry required by the core components; but if components are added, their telemetry formats need to be merged.

Starting in version 1.1.1, for situations where there is no direct GPS distribution (perhaps when the SMC is located in the SV2 sub), set **distributeGPS** to 1 and enter a command to set the date under **dateVerb** (sudo date -utc {mm}{dd}{HH}{MM}{yyyy}.{SS})

In an SV3 configuration, it is possible to receive GPS distribution via the umbilical aux lines or to receive GPS updates every second via the SV3 REST API. PayloadInterface automatically switches to reading GPS via the SV3 REST API when no NMEA stream is detected.

### PayloadInterface Commands

PayloadInterface implements the following commands, regardless of the communication protocol:

#### GETMODULES

Returns a table of the configured CORBA servers and their board addresses. For example:

```
$ ServerExec PayloadInterface GETMODULES
Address  Readable Name
-----  -
0x2000   PayloadInterface
0x20F1   GPSListener

My board address = 0x2000
Default board ID = 0x20
```

#### RUNSHELL *command-line*

Runs *command-line* as a Linux shell command and returns the output as a command result. For example:

```
$ ServerExec PayloadInterface 'RUNSHELL ls -l'
total 84
-rw-rw-r-- 1 smc smc 13922 May  9 23:13 PayloadInterface-config.xml
-rw-rw-r-- 1 smc smc  8881 May  9 02:29 PayloadInterface-telemetry-config.xml
-rw----- 1 smc smc   968 May 13 23:15 PayloadInterface.json
-rwx----- 1 smc smc   202 Apr 30 21:49 startup
```

**SendAlarmCleared** sender=<address> text=<text> {reporter=<ID>}  
{notification=<ID>} {enum=<ID>} {data=<hex string>}  
**SendAlarmSet** sender=<address> text=<text> {reporter=<ID>}  
{notification=<ID>} {enum=<ID>} {data=<hex string>}  
**SendAlarmEvent** sender=<address> text=<text> {reporter=<ID>}  
{notification=<ID>} {enum=<ID>} {data=<hex string>}  
**SendEvent** sender=<address> text=<text> {reporter=<ID>} {notification=<ID>}  
{enum=<ID>} {data=<hex string>}

These four commands allow different kinds of alarm events to be sent from the payload. The commands were included for testing: there is a CORBA message (AlarmMessage) that carries this same information; however, it is possible to implement the interface using ServerExec and these commands:

- SendAlarmSet reports an alarm in a persistent state: it cannot be cleared by the operator as long as it persists.
- SendAlarmCleared reports that a previously-reported persistent alarm is no longer asserted, and it can be completely cleared.
- SendAlarmEvent reports an alarm as a one-time event: the operator can clear it right away.
- SendEvent reports a non-alarm event.

The parameters are as follows:

- The sender address is the 16-bit board ID listed as the source of the message (required)
- The text field is the free-form text to report (required)
- The reporter ID is a 32-bit ID number that identifies a type of reporter (optional, 0 by default)
- The notification ID is a 32-bit number that identifies a kind of alarm associated with the reporter (optional, 0 by default)
- The enumerated ID is a 32-bit number that is used in combination with reporter ID and notification ID to make alarms that are set and cleared unique (optional, 0 by default)
- By convention, reporter ID values 1-999 are reserved to LRI and values 1000-1999 are reserved LRI-developed SMC software. Third-party developers should contact LRI to obtain a reserved range.

SMC services are setting the enumerated ID equal to the board address of the service, which allows alarms from difference service instances to be differentiated. For example, if there are multiple ADCP's in the same payload box, or if there are ADCP's in two different payload boxes, their alarms will not be conflated.

PayloadInterface implements the following commands only when running the SV2 payload protocol:

#### **EXECFCC** *hex-string*

Decodes *hex-string* as a sequence of bytes and re-interprets it as a Level 2 message. A CRC is added and the result is sent to the Float C&C as a command. This is a test function.

**FLOATREBOOT**

Sends a Float reboot message to the Float C&C, which causes the Float C&C software to restart.

**LIGHTOFFTEST**

Sends a "light off" command to the Float C&C, which causes the Float C&C to turn off its visibility light. This is a test function.

**READUID**

Attempts to read the unique ID from the Float C&C or VMC (the VMC does not support this at the time of this writing.)

```
$ ServerExec PayloadInterface READUID
ACK1E:23306
```

**SETPARAM** *parameter-name=new-value { parameter-name<sub>2</sub>=new-value<sub>2</sub> { ... } }*

Sends a "Set Parameters" message, with one or more parameters set to new values. The following are the parameters that can be set:

| Parameter Name(s)       | Description                                                                                                                                  | Type                          |
|-------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------|
| AIS                     | AIS Power                                                                                                                                    | Boolean (true or false)       |
| Bubble                  | Bubble Board Power                                                                                                                           | Boolean (true or false)       |
| CamCtrl                 | Camera Control                                                                                                                               | Boolean (true or false)       |
| ENUMDELAY               | Payload power-on enumeration delay (seconds)                                                                                                 | 16-bit unsigned integer       |
| FASTMB                  | Fast Mailbox check enabled (forces Iridium session every minute)                                                                             | Boolean (true or false)       |
| FTReset                 | Setting this parameter resets the state of the file transfer process; in particular, it can be used to recover from a "too many files" error | Boolean (true or false)       |
| GPSAltitudeFilter       | Determines whether GPS readings in which the altitude is suspicious are ignored                                                              | Boolean (true or false)       |
| IGain                   | Integral Gain for PID Navigation                                                                                                             | 16-bit signed integer         |
| InvalidGPSCount         | Count of invalid GPS samples (not settable)                                                                                                  | 8-bit unsigned integer        |
| iPIB                    | iPIB Power/Enable                                                                                                                            | Boolean (true or false)       |
| iPIBRESET               | Reset iPIB                                                                                                                                   | Boolean (true or false)       |
| ISatLimit               | Integral Saturation Limit for PID Navigation                                                                                                 | 16-bit unsigned integer       |
| MAGDEC                  | Magnetic Declination                                                                                                                         | 8-bit signed integer          |
| ModemOffRequiredAddress | The payload board address which is monitored and which allows the Iridium modem to be powered off as long as the board address is accessible | 16-bit unsigned integer       |
| ModemOffRequiredPort    | The payload port that is monitored and which allows the Iridium modem to be powered off as long as the payload is communicating              | 8-bit unsigned integer (1..3) |
| ModemPower              | If set and other parameters are set appropriately, the Iridium modem is turned off                                                           | Boolean (true or false)       |
| ModemPowerAtBoot        | Determines if the Iridium modem is powered on at boot (ModemPower)                                                                           | Boolean (true or false)       |
| Payload1<br>PL1         | Payload #1 Power                                                                                                                             | Boolean (true or false)       |
| Payload2<br>PL2         | Payload #2 Power                                                                                                                             | Boolean (true or false)       |
| Payload3<br>PL3         | Payload #3 Power                                                                                                                             | Boolean (true or false)       |
| PEP                     | Power Extension Port Power                                                                                                                   | Boolean (true or false)       |
| PGain                   | Proportional Gain for PID Navigation                                                                                                         | 16-bit signed integer         |

| Parameter Name(s)   | Description                                 | Type                    |
|---------------------|---------------------------------------------|-------------------------|
| PIDNav              | PID Navigation Mode (Line Following) Enable | Boolean (true or false) |
| SBDAlwaysOn         | SBD is always powered                       | Boolean (true or false) |
| SBDDefaultOn        | SBD defaults to powered on and in use       | Boolean (true or false) |
| Sub                 | Sub Power                                   | Boolean (true or false) |
| SubPayload<br>SubPL | Sub Payload Power                           | Boolean (true or false) |
| Water               | Water Speed Sensor Power                    | Boolean (true or false) |
| Weather             | Weather Station Power                       | Boolean (true or false) |
| XBEECHANNEL         | XBee modem channel number (12-23)           | 8-bit unsigned integer  |
| XBEEPANID           | XBee modem PAN ID                           | 16-bit unsigned integer |

**WCS** *address command*

Sends a Write Console Script to another payload using the given address. The output of the command is the type of ACK or NAK followed by the text output. For example, the following sends a Write Console Script to address 0 (the Float C&C) and runs its "FCC -S" command, which lists the enumeration state:

```
$ ServerExec PayloadInterface 'WCS 0 fcc -s'
ACK3D:PL1:On 0x0200(Full PIB v2.01.393) 0x2100(Sensor mgmt. compute v0.00.
001) 0x2000(Sensor mgmt. compute v2.01.000) PL2:Off PL3:Off iPIB:On 0x1000
(v1.03.350) No UnTwist
```

The following sends a Write Console Script to address 0x200 (by convention, a PIB) and runs its "PIB -C" command, which lists its configuration:

```
$ ServerExec PayloadInterface 'WCS 0x200 pib -c'
ACK3D:PM1=COMM PM2=SMC PM3=EMPTY Power Down Delay = 1 seconds
```

PayloadInterface implements the following commands only when running the SV3 REST API:

**HOSTGPS** *on-or-off*

PayloadInterface typically does not poll the VMC for GPS unless the system GPS (typically published by GPSTListener) is off-line. This command can require PayloadInterface to poll the VMC for GPS by setting HOSTGPS mode on.

**HOSTINVOKE** *{--board-address=source-address}{--hosttimeout=seconds} script-command*

"HOSTINVOKE" sends a script command to the VMC and returns the result. If a board address is specified, the command is executed as though it came from the specified address; otherwise, the command will appear to be from PayloadInterface. Note that when specifying a script command from WGMS, it must be prefixed with a percent sign ("%"); but this is not the case when using "HOSTINVOKE."

**KILLPOWER** *{--hosttimeout=seconds}*

"KILLPOWER" sends a message to the VMC to shut down power to the port. When this happens, PayloadInterface's polling commands to the VMC will return indicating that the port is about to be powered off. PayloadInterface then requests the VMC to delay shutdown according to the "powerOffDelay" value in the XML file. PayloadInterface then sends out the Shutdown CORBA event and performs an orderly system shutdown -- using shutdown(8).

**SETAMPSSW** `{--board-address=source-address} {--hosttimeout=seconds}`  
`{switch-name} on-or-off`

"SETAMPSSW" sends a message to the VMC requesting a power switch resident in the payload box to be turned on or off. One or more power switches can be associated with each payload in the JSON using the "hardware" and "connector" keys:

```
{ "boardAddress":0,
  "corbaName":"PayloadInterface",
  "description":"Payload interface",
  "hardware":[{"connector":"ICPwr1"}],
  "hostID":"1022102 SOMDM3730-20-2780AGCR-B 0213A00074",
  "interface":[{"swver":"2.1.0"}],
  "reportingAddress":8192,
```

In this case, the switch name is optional and command affects staall listed power switches. Otherwise, it affects only the indicated switch. Switches are named like "ICPwrn", where *n* ranges from 1 to the count of switches available.

Specifying a board address clarifies the payload executing the command (and the default switches associated with its registration.) If no board address is specified, the command is assumed to have come from PayloadInterface.

Note that the optional "--hosttimeout" parameter causes PayloadInterface to wait for up to the specified number of seconds for the connection to be valid before attempting to make the connection. The clientDefaultHostConnectSec setting in the XML file contains the default value if the "--hosttimeout" parameter is not specified.

#### "hostid.txt" File

PayloadInterface attempts to read the host identification strings at software startup and writes them to "hostid.txt" An SV3 currently returns strings in a format as follows:

```
{sn=1260095230, longname=Abalone, imei=300434060008890, shortname=Abalone, softwar
eChannel=V1.0.1, lavatar=/Avatar1240598335.png, avatar=http://liquidr.github.io/av
atars/Abalone.png, requires=liquidr}
```

The most significant string is the "sn" (serial number, also known as unique ID.) When working with SV2, only the serial number is written:

```
{sn=12345}
```

## PPPHost Configuration

PPPHost establishes a Point-To-Point Protocol-based internet connection via an Iridium modem periodically or upon command.

#### Common XML file keys to edit

- **scpExecutorSettings → callInInterval** contains the number of seconds between automated call-ins (default 3600, or 1 hour) If this value is set to zero (0), connections are not made on a timed interval. However, the CallNow script (or executing "ServerExec PPPHost CALLNOW") causes a connection attempt to be made immediately.
- **scpExecutorSettings → scpVerb** contains the shell script command to execute just after the connection to the Internet is made. It is initially set to `./runscp 0002`, which causes a new sub-directory (`~/roots/200_PPPHost/workspace`) to be created and a "runme" script file to be copied from an LRI file server under the 0002 directory to the new

workspace directory. The follow-up command is configured with the **runVerb** key.

- **scpExecutorSettings → runVerb** contains the shell script command to execute after successful execution of the **scpVerb** command. By default, it changes to the workspace directory and runs the "runme" file that was just copied.

Using this scheme, a shore-side server can input commands to be executed on call-in and vary the commands over time.

### One-Time Call-in Execution

When the SMC software is not running or when PPPHost is not configured to run automatically, it can be launched from the command line as follows (must start in /home/smc/roots/200\_PPPHost directory to pick up the configuration file values):

- (1) `PPPHost once`
- (2) `PPPHost once scp-verb`
- (3) `PPPHost once scp-verb run-verb`

Invocation #1 runs the normal call-in scenario, using the scp and run verbs from the PPPHost configuration file.

Invocation #2 runs the call-in scenario, but overrides the scp verb and has no run verb. This is ideal for doing a single action or running a script with the PPP session up and then immediately closing it afterward.

Invocation #3 runs the call-in scenario, but overrides both scp and run verbs.

One-time execution was implemented as an aid to scripting, but it is feasible to use it in a standalone capacity. For example, PPPHost is not launched by the startup script, but is called with the one-time execution function as needed.

### Configuring the RUDICS Modem

1. With the modem powered on and connected to an available serial port, establish a terminal session to the modem using HyperTerminal or putty (Windows) or GkTerm, minicom or screen (Linux) The default configuration should be able to operate at 19,200 baud, 8 data bits, 1 stop bit and no parity.
2. Reset the modem to factory default configuration by typing `AT&F0` (enter) and verify that the modem answers with `OK` or `0`
3. Set the baud rate to a fixed 19,200 baud by typing `AT+IPR=6,0` (enter) and verify that the modem answers with `OK` or `0`
4. Set 'verbose' mode by typing `AT&V1` (enter) and verify that the modem answers with `OK`
5. Set echo off by typing `AT&E0` (enter) and verify that the modem answers with `OK`
6. Save the configuration first typing `AT&W0` (enter) and then `AT&W1` (enter), verifying that the modem answer with `OK` after each command.
7. Cycle power to the modem and then type `AT+IPR?` (enter), verifying that it is still configured as set above (6,0)

8. Read the IMEI number from the modem by typing AT+CGSN (enter); if desired, record the resulting number for reference and compare with billing or provisioning records.
9. Read the ICC ID (Integrated Circuit Card Identification) or SIM card number by typing AT+CICCID (enter); the response should be a 19 or 20 digit number. Various errors may be returned if the SIM card is not properly installed: a SIM card is required for RUDICS. If desired, record the SIM card number for reference and compare with billing or provisioning records.

### Configure the PPP connection script

The PPP connection script (/home/smc/200\_PPPhost/runppp) uses the dialer script (above) to contact the Iridium gateway and then implements the Point-To-Point Protocol over the established connection to connect to the Internet. It contains the serial port name to be used when establishing the connection (typically /dev/ttyLX0) Edit this script to change the port name if necessary.

### Configure and verify RUDICS call parameters

Two scripts (ConfigureRUDICSCall and VerifyRUDICSCall) have been written to facilitate configuration. ConfigureRUDICSCall has several possible invocation formats:

```
(1)  ConfigureRUDICSCall key-name
(2)  ConfigureRUDICSCall key-name login-name
(3)  ConfigureRUDICSCall key-name login-name directory-name
(4)  ConfigureRUDICSCall key-name login-name directory-name
phone-number
```

The *key-name* is the name of the encryption key to be generated. Two files are generated: *key-name* and *key-name.pub*. The .pub file must be installed on the server so that ssh and scp invocations can log in without a password.

The *login-name* is the user name and the server name; for example, "PSN98765@myserver.com"

The *directory-name* is the directory that the **runscp** script goes to after logging in to get the **runme** file.

The *phone-number* is the RUDICS phone number to call; this number defaults to the number for LRI's Iridium provider (**88160000617**). If the Customer is using a different provider, the correct number should be provided.

Examples:

- ConfigureRUDICSCall PSN98765

Call into **ggsmac** server using PSN98765 as the key with the login, PSN98765@ggsmac, and copy the "runme" file from the "." directory; uses the default phone number.

```
smc@dm3730_zoom_torpedo:~$ ConfigureRUDICSCall PSN98765
Key name:      PSN98765
Phone number:  88160000617
Server login:   PSN98765@ggsmac
Server directory: .

Generating key with no pass phrase

Copy the new PSN98765.pub file to the server
```

- `ConfigureRUDICSCall PSN98765 vehicle@smactest`

Call into **smactest** server using PSN98765 as the key with the login, vehicle@smactest, and copy the "runme" file from the "PSN98765" directory; uses the default phone number.

```
smc@dm3730_zoom_torpedo:~$ ConfigureRUDICSCall PSN98765 vehicle@smactest
Key name:      PSN98765
Phone number:  88160000617
Server login:   vehicle@smactest
Server directory: PSN98765

Generating key with no pass phrase

Copy the new PSN98765.pub file to the server
```

- `ConfigureRUDICSCall PSN98765 "" "" 88169999999`

Call into **ggsmac** server using PSN98765 as the key with the login, PSN98765@ggsmac, and copy the "runme" file from the "." directory; uses a custom phone number, 88169999999.

```
smc@dm3730_zoom_torpedo:~$ ConfigureRUDICSCall PSN98765 "" "" 88169999999
Key name:      PSN98765
Phone number:  88169999999
Server login:   PSN98765@ggsmac
Server directory: .

PPPHost process is currently disabled; do you want to enable it? n
Generating key with no pass phrase

Copy the new PSN98765.pub file to the server
```

- `ConfigureRUDICSCall PSN98765 vehicle@smactest "" 88169999999`

Call into **smactest** server using PSN98765 as the key with the login, vehicle@smactest, and copy the "runme" file from the "PSN98765" directory; uses a custom phone number, 88169999999

```
smc@dm3730_zoom_torpedo:~$ ConfigureRUDICSCall PSN98765 vehicle@smactest ""
88169999999
Key name:      PSN98765
Phone number:  88169999999
Server login:   vehicle@smactest
Server directory: PSN98765

PPPHost process is currently disabled; do you want to enable it? n
Generating key with no pass phrase

Copy the new PSN98765.pub file to the server
```

Note that **ConfigureRUDICSCall** will prompt the user for action if the PPPHost process is not enabled in the configuration or if it is about to overwrite a previously-generated key; for example:

```
smc@dm3730_zoom_torpedo:~$ ConfigureRUDICSCall PSN98765
Key name:      PSN98765
Phone number:  88160000617
Server login:   PSN98765@ggsmac
Server directory: .

PPPHost process is currently disabled; do you want to enable it? y
One or more of the key files (PSN98765.pub or PSN98765) already exist; are you
sure you want to overwrite them? y
Generating key with no pass phrase
PSN98765 already exists.
Overwrite (y/n)?

Copy the new PSN98765.pub file to the server
```

**ConfigureRUDICSCall** updates the *chat-iridium* script and also writes a bash-compatible source file (.rudics) in the PPPHost configuration directory. Scripts like "runscp" or server-side scripts can include this file (source .rudics) and make use of the variables. Here's an example file:

```
KEYNAME="joe"
PHONENUM="88160000999"
SERVERLOGIN="joe@joesserver"
SERVERDIR="joesdirectory"
```

**VerifyRUDICSCall** runs PPPHost and updates the *known-hosts* file captured from the server. The log is output to the console:

```
smc@dm3730_zoom_torpedo:~/roots/200_PPPHost$ VerifyRUDICSCall

Key name:          PSN98765
Server login:       PSN98765@ggsmac
Server directory:   .

Removing existing ControlMaster, if any. Control socket
connect(/home/smc/.ssh/controlmasters/mikessmc@216.93.163.108:22): Connection
refused

2000/01/01 00:01:16.088511 dm3730_zoom_torpedo PPPHost[609] SCPExecutor
initializing
2000/01/01 00:01:16.117014 dm3730_zoom_torpedo PPPHost[609] Running the call-in
scenario once
2000/01/01 00:01:16.120615 dm3730_zoom_torpedo PPPHost[609] SCPExecutor background
thread starting
2000/01/01 00:01:16.125833 dm3730_zoom_torpedo PPPHost[609] SCPExecutor delaying 2
seconds for startup settling
2000/01/01 00:01:16.138101 dm3730_zoom_torpedo PPPHost[609] SCPExecutor delaying
1.98 seconds for startup settling
2000/01/01 00:01:18.126189 dm3730_zoom_torpedo PPPHost[609] SCPExecutor starting
to connect
2000/01/01 00:01:18.137633 dm3730_zoom_torpedo PPPHost[609] PPPHost new connection
status is DISCONNECTED was DISCONNECTED
2000/01/01 00:01:18.149840 dm3730_zoom_torpedo PPPHost[609] PPPHost new connection
status is INITIALIZING_MODEM was DISCONNECTED
SetPowerSwitch: 0x11
2000/01/01 00:01:21.807993 dm3730_zoom_torpedo PPPHost[609] PPPHost to send
command 'ATV1' expecting 'OK' if successful or 'ERROR' for errors
2000/01/01 00:01:21.811015 dm3730_zoom_torpedo PPPHost[609] PPPHost sending
'ATV1\r' to modem
2000/01/01 00:01:21.822703 dm3730_zoom_torpedo PPPHost[609] PPPHost expecting a
single line within 5 seconds
...
2000/01/01 00:01:38.586500 dm3730_zoom_torpedo PPPHost[609] Executing protocol
daemon verb './runppp'
2000/01/01 00:01:38.600660 dm3730_zoom_torpedo PPPHost[609]
AsynchronousProcessExecutor background thread started
2000/01/01 00:01:39.136149 dm3730_zoom_torpedo PPPHost[609] Daemon> abort on
(BUSY)
2000/01/01 00:01:39.140147 dm3730_zoom_torpedo PPPHost[609] Daemon> abort on (NO
CARRIER)
2000/01/01 00:01:39.140787 dm3730_zoom_torpedo PPPHost[609] Daemon> abort on
(ERROR)
2000/01/01 00:01:39.143503 dm3730_zoom_torpedo PPPHost[609] Daemon> abort on (NO
DIALTONE)
...
2000/01/01 00:01:39.995306 dm3730_zoom_torpedo PPPHost[609] Daemon> send
(ATDT0088160000617^M)
2000/01/01 00:01:40.184483 dm3730_zoom_torpedo PPPHost[609] Daemon> expect
(CONNECT)
2000/01/01 00:01:40.189061 dm3730_zoom_torpedo PPPHost[609] Daemon> ^M
2000/01/01 00:01:51.661076 dm3730_zoom_torpedo PPPHost[609] Daemon> ^M
2000/01/01 00:01:51.668919 dm3730_zoom_torpedo PPPHost[609] Daemon> CONNECT
2000/01/01 00:01:51.673375 dm3730_zoom_torpedo PPPHost[609] Daemon> -- got it
```

```
2000/01/01 00:01:51.676274 dm3730_zoom_torpedo PPPHost[609] Daemon>
2000/01/01 00:01:51.679021 dm3730_zoom_torpedo PPPHost[609] Daemon> send (\d)
2000/01/01 00:01:52.687438 dm3730_zoom_torpedo PPPHost[609] Daemon> Script
/usr/sbin/chat -s -v -t 240 -f chat-iridium finished (pid 629), status = 0x0
2000/01/01 00:01:52.689147 dm3730_zoom_torpedo PPPHost[609] Daemon> Serial
connection established.
2000/01/01 00:01:52.696441 dm3730_zoom_torpedo PPPHost[609] Daemon> using channel
1
2000/01/01 00:01:52.751799 dm3730_zoom_torpedo PPPHost[609] Daemon> Using
interface ppp0
2000/01/01 00:01:52.766082 dm3730_zoom_torpedo PPPHost[609] Daemon> Connect: ppp0
<--> /dev/ttyLX0
2000/01/01 00:01:52.768737 dm3730_zoom_torpedo PPPHost[609] PPPHost new connection
status is CONNECTING was DIALING
2000/01/01 00:01:53.776452 dm3730_zoom_torpedo PPPHost[609] Daemon> Warning -
secret file /etc/ppp/pap-secrets has world and/or group access
2000/01/01 00:01:53.788781 dm3730_zoom_torpedo PPPHost[609] Daemon> sent [LCP
ConfReq id=0x1 <asynmap 0x0> <magic 0x87dlf4f7> <pcomp> <accomp>]
2000/01/01 00:01:55.000292 dm3730_zoom_torpedo PPPHost[609] Daemon> rcvd [LCP
ConfReq id=0x41 <asynmap 0xa0000> <auth chap MD5> <magic 0x7a8fe513> <pcomp>
<accomp> <mrru 1524> <endpoint [local:4d.4d.4c.50.5f.73.74.61.63.6b]>]
...
2000/01/01 00:02:04.224761 dm3730_zoom_torpedo PPPHost[609] Daemon> local IP
address 192.168.20.150
2000/01/01 00:02:04.229797 dm3730_zoom_torpedo PPPHost[609] Daemon> remote IP
address 192.168.20.254
2000/01/01 00:02:04.234069 dm3730_zoom_torpedo PPPHost[609] Waiting for 1 second
after PPP connected
2000/01/01 00:02:04.240356 dm3730_zoom_torpedo PPPHost[609] Daemon> primary DNS
address 199.2.252.10
2000/01/01 00:02:04.244933 dm3730_zoom_torpedo PPPHost[609] Daemon> secondary DNS
address 204.97.212.10
2000/01/01 00:02:04.267333 dm3730_zoom_torpedo PPPHost[609] Daemon> Script
/etc/ppp/ip-up started (pid 630)
2000/01/01 00:02:04.472776 dm3730_zoom_torpedo PPPHost[609] Daemon> Script
/etc/ppp/ip-up finished (pid 630), status = 0x0
2000/01/01 00:02:05.242212 dm3730_zoom_torpedo PPPHost[609] PPPHost new connection
status is CONNECTED was CONNECTING
2000/01/01 00:02:05.245325 dm3730_zoom_torpedo PPPHost[609] Saving modem serial
number '300025010413330' to IMEI
2000/01/01 00:02:05.254327 dm3730_zoom_torpedo PPPHost[609] Saving modem SIM card
number '8988169414001281532' to SIMCARD
2000/01/01 00:02:05.261072 dm3730_zoom_torpedo PPPHost[609] SCPExecutor running
SCP verb 'ssh -o StrictHostKeyChecking=no -i PSN98765 PSN98765@ggsmac "echo
\"Reply from server\"" ; sleep 2'
2000/01/01 00:02:05.265466 dm3730_zoom_torpedo PPPHost[609]
AsynchronousProcessExecutor background thread started
2000/01/01 00:02:26.597629 dm3730_zoom_torpedo PPPHost[609] Received SCP_OUTPUT
'Warning: Permanently added '216.93.163.108' (RSA) to the list of known hosts.'
2000/01/01 00:02:46.746448 dm3730_zoom_torpedo PPPHost[609] Received SCP_OUTPUT
'Reply from server'
2000/01/01 00:02:49.719455 dm3730_zoom_torpedo PPPHost[609]
AsynchronousProcessExecutor waitpid(634): exited with status 0
2000/01/01 00:02:49.722018 dm3730_zoom_torpedo PPPHost[609] SCP_EXITED exited with
status 0
2000/01/01 00:02:49.723758 dm3730_zoom_torpedo PPPHost[609] SCP command succeeded
...
2000/01/01 00:03:01.783295 dm3730_zoom_torpedo PPPHost[609] PPPHost new connection
status is DISCONNECTED was DISCONNECTING
2000/01/01 00:03:01.786682 dm3730_zoom_torpedo PPPHost[609] PPPHost new connection
status is DISCONNECTED was DISCONNECTED (not firing event)
SetPowerSwitch: 0x00
2000/01/01 00:03:02.400326 dm3730_zoom_torpedo PPPHost[609] SCPExecutor starting
to shut down
2000/01/01 00:03:02.401425 dm3730_zoom_torpedo PPPHost[609] Executing next call
interval verb 'if [ -f callinTime ] ; then cat callinTime ; fi'
2000/01/01 00:03:02.451351 dm3730_zoom_torpedo PPPHost[609] Next call interval
verb returned blank
2000/01/01 00:03:02.452389 dm3730_zoom_torpedo PPPHost[609] Next call interval is
3600 seconds
```

```
2000/01/01 00:03:02.453274 dm3730_zoom_torpedo PPPHost[609] SCPExecutor delaying
3600 seconds for call-in interval
2000/01/01 00:03:02.457150 dm3730_zoom_torpedo PPPHost[609] SCPExecutor thread
received shutdown request and will now exit (SCPExecutor detected object shutdown
while delaying for call-in interval)
2000/01/01 00:03:02.457913 dm3730_zoom_torpedo PPPHost[609] SCPExecutor thread
exiting
2000/01/01 00:03:02.459957 dm3730_zoom_torpedo PPPHost[609]
2000/01/01 00:03:02.460690 dm3730_zoom_torpedo PPPHost[609]
2000/01/01 00:03:02.461392 dm3730_zoom_torpedo PPPHost[609] ***
SUCCEEDED
```

The final line should indicate that the operation succeeded. The listing above also shows that the key from the server was saved.

### PPPHost Commands

The following CORBA commands are supported by PPPHost, so they can be executed using ServerExec or via PayloadInterface:

#### CALLNOW

If there is no call-in session executing, CallNow schedules one to begin immediately. This is the primary way that processes generating data to be transmitted to a server can force that action to occur.

#### CALLNOWOVERRIDEGPS

CALLNOWOVERRIDEGPS combines the function of OVERRIDEGPS and CALLNOW: a RUDICS session is scheduled immediately, regardless of whether GPS has been received.

#### GETKEY

Returns the name of the key file that is in use, which is also the server account name, by convention.

#### OVERRIDEGPS

Normally, PPPHost waits to start receiving GPS before it attempts a call. This was implemented to prevent calls from failing when the system time is set due to the first GPS reading being received. OVERRIDEGPS instructs PPPHost to ignore the fact that no GPS has been received for the current run of PPPHost (the override is not persistent.)

### Manifest File Metadata

When the payload is connected over an SV3 Expansion Port, the protocol with the VMC is to send a small JSON fragment that lists the commands that are supported by each server. The fragment also allows other information about the device to be included. The payload's JSON fragments are included in the .TXP file attached to the vehicle in WGMS.

Two sections of the JSON fragment include interesting keys:

- **RUDICS/KEYNAME** is the SSH key in use for the RUDICS connection, which by convention is also the account name used to sign in.
- **RUDICS/PHONENUM** is the phone number used to dial the gateway (useful if troubleshooting connection problems)
- **RUDICS/SERVERDIR** is the directory on the server to start in when transferring files

- **RUDICS/SERVERLOGIN** is the account name and server name used when connecting to the target over RUDICS
- **hardware[0]** contains information about the payload Iridium modem; **serial** key includes both the IMEI number and the SIM card number

```
{
  "RUDICS" : {
    "KEYNAME" : "mikessmc",
    "PHONENUM" : "88160000617",
    "SERVERDIR" : ".",
    "SERVERLOGIN" : "mikessmc@ggsmac"
  },
  "hardware" : [
    {
      "connector" : "-",
      "manufacturer" : "Iridium",
      "model" : "9522B",
      "serial" : "IMEI:300025010413330 SIM:8988169414001281532",
      "tty" : "/dev/ttyLX0"
    }
  ],
}
```

### Route Configurations

The network environment on the SMC can be complex: sometimes there are multiple networks active to support PPP, chat with the VMC over Ethernet or PPP and use of a debug cable. Starting in version 2.5.0, servers to be accessed over PPP to shore *must have specific routes configured*.

The file to be modified is `/etc/ppp/ip-up`. Add entries to the highlighted section to support any servers to be accessed over `ppp0` (the interface controlled by `PPPHost`):

```
#!/bin/sh

echo >> /etc/ppp/mylog.txt `date` UP 1=$1 2=$2 3=$3 4=$4 5=$5 6=$6 usepeerdns=$
USEPEERDNS dns1=$DNS1 dns2=$DNS2"

IFACE="$1"
SERIAL_PORT="$2"
BAUD_RATE="$3"
LOCAL_ADDR="$4"
REMOTE_ADDR="$5"

if [ "$USEPEERDNS" = "1" ]; then
  ( if [ -n "$DNS1" ]; then echo "nameserver $DNS1"; fi
    if [ -n "$DNS2" ]; then echo "nameserver $DNS2"; fi ) | /sbin/resolvconf -a
"$IFACE"
fi

if [ "$IFACE" = "ppp0" ]; then
  # add route to ggsmac
  /sbin/route add 216.93.163.108 gw "$REMOTE_ADDR" dev "$IFACE"
  # add route to smactest
  /sbin/route add 12.156.66.48 gw "$REMOTE_ADDR" dev "$IFACE"
elif [ "$IFACE" = "ppp1" ]; then
  if /sbin/route -n | grep '^0[.]0[.]0[.]0.*eth0$' | grep eth0 > /dev/null; then
    /sbin/route del default dev eth0
  fi
  /sbin/route add default gw ${REMOTE_ADDR} dev "$IFACE"
fi
```

### Tips for Script Developers

The normal scenario that `PPPHost` executes after making a connection is to read a script from a remote server and then execute it. Here are some tips for writing that script:

- The script can write a “**callInTime**” file under /home/smc/roots/200\_PPPhost with the number of seconds to wait before attempting another call. For example:
  - Write callInTime with a small callback time (let’s say 600 seconds, 10 minutes) at the start of your script
  - Write callInTime with a larger value or delete it just before exiting if there were no communication errors
- We typically use the **timelimit2** utility whenever making network requests over RUDICS to ensure that the command will exit if it’s taking too long (prevent sitting there with a dead connection)
- Starting in version 2.4.0, the scripts that come with PPPhost were modified to use the ssh -O feature to keep the connection alive between invocations of ssh and scp. This greatly decreases the startup time for each ssh/scp command because the connection does not have to be re-negotiated. Keep this in mind.
- Now that the connection is kept alive between operations, consider using **rsync** to update files.
- PayloadInterface requests information from the Float C&C or VMC and stores it in /home/smc/roots/100\_PayloadInterface. This contains identifying data about the host platform. On SV2 systems, this is only the serial number; but on SV3, it includes the IMEI number of the VMC’s Iridium modem, as well:

```
{sn=1260095230, longname=Abalone, imei=300434060008890,
shortname=Abalone, softwareChannel=V1.0.1,
lavatar=/Avatar1240598335.png,
avatar=http://liquidr.github.io/avatars/Abalone.png,
requires=liquidr}
```
- PPPhost drops several files in the /home/smc/roots/200\_PPPhost directory while making the connection that can be referenced from the script
  - **GPSLatitude** contains the current latitude of the Wave Glider (degrees)
  - **GPSLongitude** contains the current longitude of the Wave Glider (degrees)
  - **IMEI** contains the IMEI number of the RUDICS modem
  - **SIMCARD** contains the SIM card number of the RUDICS modem

## SeabirdGPCTD Configuration

SeabirdGPCTD implements an interface to a Seabird Glider Payload CTD, with or without an attached oxygen sensor. Its primary features are:

- A command interface that allows the user to start or stop sampling, request status, send low diagnostics commands and configure logging and startup actions.
- On-board log of samples and commands and responses.
- Parameterized sampling scenario that specifies how samples are concatenated into telemetry reports, sleep time between reports and input line flush times before starting to sample.

The first instance of the program starts up under **/home/smc/roots/200\_SBGPC TD**

The static configuration file is **SeabirdGCPTD-config.xml**; its salient configuration settings are:

- (1) The CORBA server name and type. As with every service, the server name must be unique, and the name is used whenever executing command via ServerExec:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!DOCTYPE boost_serialization>
<boost_serialization signature="serialization::archive" version="10">
<rootlevel class_id="0" tracking_level="0" version="1">
  <sensorProgramSettings class_id="1" tracking_level="0" version="1">
    <serverName>SBGPCTD</serverName>
    <serverKind></serverKind>
```

- (2) The board ID and task ID, which make up the board address used externally (from WGMS or the SV3 web page) to send commands to the process, and the address stamped onto telemetry samples. It is recommended to set the board ID to 255, which means that the board ID configured in PayloadManager is used. The readable name should match the server name.

```
<payloadManagerClientSettings class_id="11" tracking_level="0" version="0">
  <serviceName>PayloadManager</serviceName>
  <serviceKind></serviceKind>
  <deviceAddedEventName>PayloadDeviceAdded</deviceAddedEventName>
  <deviceChangedEventName>PayloadDeviceChanged</deviceChangedEventName>
  <deviceRemovedEventName>PayloadDeviceRemoved</deviceRemovedEventName>
  <commonEventType>PayloadRegistrationEvent</commonEventType>
  <commonEventDomain>LiquidR-SMC</commonEventDomain>
  <payloadDescription class_id="12" tracking_level="0" version="2">
    <boardID>255</boardID>
    <taskID>38</taskID>
    <description>Seabird GPCTD interface</description>
    <ior></ior>
    <readableName>SBGPCTD</readableName>
    <descriptiveJSON></descriptiveJSON>
  </payloadDescription>
</payloadManagerClientSettings>
```

- (3) The sensor is powered on and off using the powerOnVerb and the powerOffVerb. Use **sudo SetPowerSwitch switch-number new-state** to affect the state of the local power outputs, or substitute other commands<sup>1</sup> to affect appropriate power switch.

```
<sensorSettings class_id="14" tracking_level="0" version="1">
  <powerOffVerb>sudo SetPowerSwitch 3 0</powerOffVerb>
  <powerOnVerb>sudo SetPowerSwitch 3 1</powerOnVerb>
  <serialListenerSettings class_id="15" tracking_level="0" version="0">
    <maxReadSize>1024</maxReadSize>
    <intercharacterGapTimeoutMs>50</intercharacterGapTimeoutMs>
  </serialListenerSettings>
  <serialPortSettings class_id="16" tracking_level="0" version="0">
    <portName>/dev/ttyLX6</portName>
    <baudRate>115200</baudRate>
    <parity>NONE</parity>
    <characterSize>8</characterSize>
    <stopBits>1</stopBits>
    <flowControl>NONE</flowControl>
    <rts>1</rts>
```

<sup>1</sup> For example, PayloadInterface implements some commands to affect AMPS power switches for SV3-style payloads (**SETAMPSSW**) or to set power switches using runtime parameters from SV2-style payloads (**SETPARAM.**) These can be executed using ServerExec.

```

        <dt>1</dt>
        <breakTimeoutMultiplier>1</breakTimeoutMultiplier>
    </serialPortSettings>
</sensorSettings>

```

The serial port device name (portName) and the baud rate to be used (baudRate) can also be set.

- (4) There is a setting for the logger that should be modified if there are additional instances. Specifically, the directory where the logs are stored should be changed to prevent output from different sensor instances from getting intermixed or overwritten:

```

<fileSetSettings class_id="14" tracking_level="0" version="0">
    <archiveCompressionType>BZIP2</archiveCompressionType>
    <archiveDirectory></archiveDirectory>
    <extension>.CSV</extension>
    <keepWritesTogether>1</keepWritesTogether>
    <maxBytesQueued>10000</maxBytesQueued>
    <maxFileSize>1000000</maxFileSize>
    <maxFiles>1000</maxFiles>
    <maxRetainTime>00:00:00</maxRetainTime>
    <prefix>SBGPCTD</prefix>
    <purgeStrategy>OLDEST</purgeStrategy>
    <workingDirectory>/mnt/XDATA/SBGPCTD</workingDirectory>
</fileSetSettings>

```

The persistent configuration file, which holds values that can be changed through Write Console Script or ServerExec commands, is **SeabirdGCPTD-persist-config.xml**:

```

<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!DOCTYPE boost_serialization>
<boost_serialization signature="serialization::archive" version="10">
<rootlevel class_id="0" tracking_level="0" version="1">
    <autoRunStr>false</autoRunStr>
    <dutyCycleSec>0</dutyCycleSec>
    <flushTimeSec>5</flushTimeSec>
    <loggingEnabledStr>true</loggingEnabledStr>
    <rawLoggingEnabledStr>false</rawLoggingEnabledStr>
    <reportBlockSize>8</reportBlockSize>
    <samplePeriodSec>60</samplePeriodSec>
</rootlevel>
</boost_serialization>

```

Note that the settings in the persistent configuration XML file are tracked with an MD5 signature file to detect corruption, **so this file cannot be edited by hand**. In the case where the MD5 signature doesn't match, all settings return to default values (as above.)

### SeabirdGPCTD Telemetry Output

Two types of telemetry are output under payload type 0x70:

1. Telemetry sub-type 8 as CORBA message name "SBGPCTDSample" for samples with no O2 sensor installed
2. Telemetry sub-type 9 as CORBA message name "SBGPCTDSampleWithDO" for samples with O2 sensor installed.

The program is operating the sensor with real-world units, but types 8 and 9 assume that the sensor is reporting in engineering units. As a result, the equivalent engineering units are computed and both sets of values are included in the CORBA event:

| Tag                           | Description                                      | Type   |
|-------------------------------|--------------------------------------------------|--------|
| SBGPCTDSample_ConductivityStr | The conductivity reading from the sensor exactly | STRING |

| Tag                              | Description                                                         | Type         |
|----------------------------------|---------------------------------------------------------------------|--------------|
|                                  | as output (S/m)                                                     |              |
| SBGPCTDSample_RawConductivity    | Conductivity (Engineering units, use this for telemetry)            | INT32        |
| SBGPCTDSample_Conductivity       | Conductivity string converted to float                              | SINGLE_FLOAT |
| SBGPCTDSample_TemperatureStr     | The temperature reading from the sensor exactly as output (C)       | STRING       |
| SBGPCTDSample_RawTemperature     | Temperature (Engineering units, use this for telemetry)             | INT32        |
| SBGPCTDSample_Temperature        | Temperature string converted to float                               | SINGLE_FLOAT |
| SBGPCTDSample_RawPressure        | Pressure (Engineering units, use this for telemetry)                | INT32        |
| SBGPCTDSample_PressureStr        | The pressure reading from the sensor exactly as output (decibars)   | STRING       |
| SBGPCTDSample_Pressure           | Pressure string converted to float                                  | SINGLE_FLOAT |
| SBGPCTDSample_DissolvedOxygenStr | The dissolved oxygen reading from the sensor exactly as output (Hz) | STRING       |
| SBGPCTDSample_RawDissolvedOxygen | Dissolved oxygen (Engineering units, use this for telemetry)        | INT32        |
| SBGPCTDSample_DissolvedOxygen    | Dissolved oxygen string converted to float                          | SINGLE_FLOAT |
| SBGPCTDSample_O2Installed        | True if an O2 sensor is installed, false if it is not installed     | BOOLEAN      |

The following is the XML configuration to insert into PayloadInterface-telemetry-config.xml to convert the sensor reading to binary. Note that the number of samples per report will be modified by the runtime configuration of the sensor.

```

<Element>
  <first boardID="0xFF" taskID="0x26" />
  <second>
    <key boardID="0xFF" taskID="0x26" />
    <payloadType value="0x00000070" />
    <eventNameToFormatCode>
      <Element>
        <first value="SBGPCTDSample" />
        <second value="8" />
      </Element>
      <Element>
        <first value="SBGPCTDSampleWithDO" />
        <second value="9" />
      </Element>
    </eventNameToFormatCode>
    <reportParams>
      <Element>
        <first value="8" />
        <second maxSamples="1" targetAddress="2">
          <reportHeaderFormat>
            <fields>
              <Element libField="Header91" />
              <Element libField="Raw" eventField="Telem-
LengthToFollow" />
              <Element libField="Raw" eventField="Telem-PayloadType"
/>
              <Element libField="Raw" eventField="Telem-BoardID" />
              <Element libField="Raw" eventField="Telem-TaskID" />
              <Element libField="FC8BinaryNormal" />
              <Element libField="RawINT8" eventField="Telem-
NumSamples" />
            </fields>
          </reportHeaderFormat>
          <sampleHeaderFormat>
            <fields>
              <Element libField="Raw" eventField="Telem-Latitude" />
              <Element libField="Raw" eventField="Telem-Longitude" />
              <Element libField="Raw" eventField="Telem-TimeSec" />
            </fields>
          </sampleHeaderFormat>
        </second>
      </Element>
    </reportParams>
  </second>
</Element>

```

```

                                <Element libField="Raw"
eventField="SBGPCTDSample_RawPressure" />
                                <Element libField="Raw"
eventField="SBGPCTDSample_RawTemperature" />
                                <Element libField="Raw"
eventField="SBGPCTDSample_RawConductivity" />
                                </fields>
                                </sampleHeaderFormat>
                                </second>
                                </Element>
                                <Element>
                                  <first value="9" />
                                  <second maxSamples="1" targetAddress="2">
                                    <reportHeaderFormat>
                                      <fields>
                                        <Element libField="Header91" />
                                        <Element libField="Raw" eventField="Telem-
LengthToFollow" />
                                        <Element libField="Raw" eventField="Telem-PayloadType"
/>
                                        <Element libField="Raw" eventField="Telem-BoardID" />
                                        <Element libField="Raw" eventField="Telem-TaskID" />
                                        <Element libField="FC8BinaryNormal" />
                                        <Element libField="RawINT8" eventField="Telem-
NumSamples" />
                                      </fields>
                                    </reportHeaderFormat>
                                    <sampleHeaderFormat>
                                      <fields>
                                        <Element libField="Raw" eventField="Telem-Latitude" />
                                        <Element libField="Raw" eventField="Telem-Longitude" />
                                        <Element libField="Raw" eventField="Telem-TimeSec" />
                                        <Element libField="Raw"
eventField="SBGPCTDSample_RawPressure" />
                                        <Element libField="Raw"
eventField="SBGPCTDSample_RawTemperature" />
                                        <Element libField="Raw"
eventField="SBGPCTDSample_RawConductivity" />
                                        <Element libField="Raw"
eventField="SBGPCTDSample_RawDissolvedOxygen" />
                                      </fields>
                                    </sampleHeaderFormat>
                                    </second>
                                  </Element>
                                </reportParams>
                                </second>
                                </Element>

```

### SeabirdGPCTD Commands

SeabirdGPCTD implements the following commands:

#### **ctd --autorun <on-or-off>**

Enables (on/true) or disabled (off/false) logging of all messages and responses from the sensor.

#### **ctd --expert <command>**

Send one of the supported device commands and get the output from it. The supported commands are:

- dc (display calibration coefficients)
- ds (display status)
- getcc (get calibration coefficients)
- getcd (get configuration data: XML)

getec (get event counters: XML)  
gethd (get hardware configuration: XML)  
getsd (get status data: XML)  
sl (send last sample)  
slp (send last pressure sample)

### **ctd --power <on-or-off>**

Enables (on/true) or disabled (off/false) power to the sensor. Note that power is automatically turned on when sampling is started, so there's no need to send an additional command to power it on.

### **ctd --start {<period(sec)> {<samples-per-block> {<flush-time(sec)> {<off-time(sec)>}}}}**

Turns on power to the sensor and starts taking readings, returning telemetry for each sample. If included, the parameters overwrite the currently-configured values (the new values are saved persistently.)

- period(sec) is the number of seconds between samples. Note that this value cannot be over 14 seconds if an O2 sensor is installed (see GPCTD manual for further detail.)
- samples-per-block is the number of samples per telemetry sample, and number of samples between imposing an off time.
- flush-time(sec) is the number of seconds to run the pump (to get representative material in the line) before starting to sample.
- off-time(sec) is the number of seconds to suspend pumping and sampling between blocks. If this value is zero, sampling is continuous.

### **ctd --status**

Returns a message representing the state of the sensor. For example:

```
$ ServerExec SBGPCTD "ctd --status"  
OK Sampling Port=/dev/ttyLX6 Enable=true Pwr=true Intervl=1 sec Size=10 smpl/pkt  
Off-Tm=5 sec Flush-Tm=3 Oxygen=true AutoRun=true Logging=samples
```

### **ctd --stop**

Stops the sampling loop, but leaves the sensor powered (in low power state.)

## **SimpleSBDQueue Configuration**

SimpleSBDQueue is a service that interfaces an Iridium 9522B or 9523 SBD (Short Burst Data) satellite modem and connects it to the SMC's CORBA infrastructure. Messages sent to the service are stored in a queue and are sent sequentially as SBD messages as fast as the policy allows; messages received over SBD are sent as CORBA events.

### **Commands**

SimpleSBDQueue supports the following commands:

#### **SSQ --run <filename>**

Opens *filename* and runs any commands it contains.

**SSQ --enqueue [--priority=<priority>] [--senderid=<senderid>] [--sendertransaction=<sendertransaction>] [--queue=<position>] [--ttl=<timetolive>] <message>**

Queues up a message to be sent out over the satellite network. The content of the SBD buffer is provided by *message*, which is a hexadecimal string representing the bytes. Other variables that may be provided are:

- *priority* (unsigned 16-bit number), which represents how important the message is; messages with lower priority numbers are sent sooner. The default value is **3**.
- *senderid* (unsigned 16-bit number), which represents the board address of the sending process. Status outputs related to this message include the *senderid*. The default value is **0xFFFF**.
- *sendertransaction* (unsigned 32-bit number), which is meant to allow the sender to distinguish between sent messages. Status outputs related to this message include the *sendertransaction*. The default value is **0xFFFFFFFF**.
- *position* is where to place the message in the queue. Valid values are:
  - end: store the message at the end of the list
  - front: store the message so that it will be transmitted next
  - latest: the message will be sent next unless another message comes in, in which case it goes to the end of the queue
 The default value for *position* is "end"
- *timetolive* (unsigned 32-bit number) is how long to let the message sit in the queue before timing it out (milliseconds, 0=no timeout.) The default value is **0**.

### SSQ --mbcheck

Triggers a mailbox check, which causes the modem to be activated and an empty transmission to be sent. This has the effect of checking the status of the "mailbox" and reading the next message if any are pending. This is not typically needed, but can be useful for debugging.

### SSQ --start

Starts the background execution of the protocol, which periodically checks for messages from the satellite network and sends any pending messages.

### SSQ --stop

Stops the background protocol. No attempt will be made to check for messages from the satellite network and no messages will be sent.

## CORBA Events

SimpleSBDQueue reports the following CORBA events:

| EventName                                                                                                                                                        | TypeName               | DomainName | Class              |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|------------|--------------------|
| SBDTransmitDisposition                                                                                                                                           | SBDTransmitDisposition | LiquidR    | SBDPacketEvent     |
| This event is sent after each transmission to indicate the MOMSN and status of the transmission, including the original sender ID and transaction for reference. |                        |            |                    |
| SBDStatsUpdate                                                                                                                                                   | SBDChannelStats        | LiquidR    | SBDStatisticsEvent |

| EventName                                                                                                                | TypeName       | DomainName | Class          |
|--------------------------------------------------------------------------------------------------------------------------|----------------|------------|----------------|
| This event is sent after each connection attempt to indicate the status, success rate and reliability of the connection. |                |            |                |
| SBDReceivedPacket                                                                                                        | SBDPacketEvent | LiquidR    | SBDPacketEvent |
| This event is sent whenever a packet is received from the SBD modem.                                                     |                |            |                |

SimpleSBDQueue can receive the following CORBA events:

| EventName                                                                                           | TypeName       | DomainName | Class          |
|-----------------------------------------------------------------------------------------------------|----------------|------------|----------------|
| SBDEnqueuePacket                                                                                    | SBDPacketEvent | LiquidR    | SBDPacketEvent |
| This event can be sent to SimpleSBDQueue as alternative way to queue up a message for transmission. |                |            |                |

### XML Configuration

The configuration file (SimpleSBDQueue-config.xml), which must be located in the startup directory (typically /home/smc/roots/200\_SimpleSBDQueue), conforms to the format of other configuration files, but here are a few items of interest:

```
<autoStart>true</autoStart>
```

Configures whether to automatically begin operating on startup.

```
<concatenateSentPackets>false</concatenateSentPackets>
```

Configures whether to concatenate messages to maximize use of the buffer before transmitting.

```
<minOffTimeMS>3000</minOffTimeMS>
<maxOnTimeMS>300000</maxOnTimeMS>
```

Configures the minimum amount of time that the modem must remain off when it is turned off (*minOffTimeMS*, milliseconds) and the maximum amount of time that it can remain on (*maxOnTimeMS*, milliseconds.)

```
<dutyCycleModem>true</dutyCycleModem>
```

Configures whether modem power should be cycled on and off to reduce power usage (true) or whether it should remain on all the time (false.)

```
<networkAvailableTimeoutMS>60000</networkAvailableTimeoutMS>
<networkFailDelayMS>120000</networkFailDelayMS>
```

*networkAvailableTimeoutMS* configures how long (milliseconds) to wait for the satellite network to be available before declaring a network failure. If a network failure is declared, *networkFailDelayMS* configures how long (milliseconds) to wait before attempting communication again.

```
<mailboxCheckIntervalMS>60000</mailboxCheckIntervalMS>
```

Configures how frequently (milliseconds of idle delay) to check for messages from the satellite network. Note that if messages are being sent to the satellite network more frequently, mailbox checks will not be performed.

```
<commFailureDelayMS>120000</commFailureDelayMS>
```

If a communication failure with the modem is declared, *commFailureDelayMS* configures how long to wait (milliseconds) before attempting further communication.

```
<useRingIndicator>false</useRingIndicator>
```

Determines whether ring indication should be used to indicate when a message is waiting. Note that this is only effective if the modem is left turned on. **Note: waiting for ring indicator has not been implemented yet.**

```
<networkTimeoutSeconds>1200</networkTimeoutSeconds>
```

If the modem cannot contact the satellite network for *networkTimeoutSeconds* or more, modem communication is declared to be unreliable (to client applications.)

## TempPress Configuration

TempPress is a service that interfaces a temperature and pressure sensor and produces telemetry with the intention of eliminating the need for either a PIB or an AMPS domain, each of which has this function. Pilots use this information as a leak indication for the payload box.

The Measurement Specialties MS5803-14BA is interfaced through the I<sup>2</sup>C bus; it reports temperatures in the range of -40C to 85C and pressures in the range of 0 to 14 bars.

The primary features of the interface are:

- Commands are exposed via the SV3 Commands window
- The sensor is polled according to a "seconds" divisor with respect to elapsed time since midnight. For example, setting a divisor to 300 seconds (5 minutes) means that samples are taken on the hour, 5 minutes after the hour, 10 minutes after the hour, etc
- There are separate *reporting* and *non-reporting divisors*, each of which generates CORBA events. Other CORBA servers can be written to consume these events and use temperature and pressure in their own calculations; events associated with the *reporting divisor* result in telemetry being transmitted
- Settable limits for temperature (1C to 45C by default) and pressure (105 to 130 kPa by default) are used to generate alarms
- A human-readable status message can be polled, which includes the last reading, limits and runtime state
- Sensor data is logged to a directory in the XDATA partition
- Operational parameters can be updated via Write Console Script commands and are stored persistently

## Server Run Modes

TempPress can be invoked from the Linux command line as follows:

```
TempPress {-c|--continuous} {-g|--} {-m|--model model-spec} {-s|--server}  
{-v|--verbose}
```

The most common configuration will be to run as a CORBA server; note that this requires omniORB, EventManager and PayloadManager to be up and running already:

```
$ TempPress  
$ TempPress -s  
$ TempPress --server
```

The program can be invoked as follows to take a single reading, either in terse mode or verbose mode:

```
$ TempPress -g  
18.83C 101.88kPA @2015-09-30 01:13:21.404422  
$ TempPress --single  
18.87C 101.88kPA @2015-09-30 01:13:26.595402  
$ TempPress -gv
```

```
18.89C 101.85kPA @2015-09-30 01:13:44.190102 D1=4195780 D2=8515826 C0=0
C1=45543 C2=40489 C3=28041 C4=26448 C5=32801 C6=28264 C7=8
model=MS5803_14BA device=/dev/i2c-2 address=118
$ TempPress --single --verbose
18.88C 101.98kPA @2015-09-30 01:13:55.575093 D1=4196406 D2=8515500 C0=0
C1=45543 C2=40489 C3=28041 C4=26448 C5=32801 C6=28264 C7=8
model=MS5803_14BA device=/dev/i2c-2 address=118
```

Note that verbose mode returns raw sensor readings (D1, D2) and calibration values (C0-C7.)

The sensor can also be polled continuously:

```
$ TempPress -cv
18.84C 101.88kPA @2015-09-30 01:19:16.824680 D1=4195944 D2=8514574 C0=0
C1=45543 C2=40489 C3=28041 C4=26448 C5=32801 C6=28264 C7=8
model=MS5803_14BA device=/dev/i2c-2 address=118
18.83C 101.89kPA @2015-09-30 01:19:17.824843 D1=4196036 D2=8514044 C0=0
C1=45543 C2=40489 C3=28041 C4=26448 C5=32801 C6=28264 C7=8
model=MS5803_14BA device=/dev/i2c-2 address=118
18.85C 101.85kPA @2015-09-30 01:19:18.824825 D1=4195814 D2=8514852 C0=0
C1=45543 C2=40489 C3=28041 C4=26448 C5=32801 C6=28264 C7=8
model=MS5803_14BA device=/dev/i2c-2 address=118
$ TempPress --continuous --verbose
18.79C 102.03kPA @2015-09-30 01:19:28.160454 D1=4196666 D2=8512980 C0=0
C1=45543 C2=40489 C3=28041 C4=26448 C5=32801 C6=28264 C7=8
model=MS5803_14BA device=/dev/i2c-2 address=118
18.83C 101.72kPA @2015-09-30 01:19:29.160739 D1=4195240 D2=8514212 C0=0
C1=45543 C2=40489 C3=28041 C4=26448 C5=32801 C6=28264 C7=8
model=MS5803_14BA device=/dev/i2c-2 address=118
18.83C 101.73kPA @2015-09-30 01:19:30.160629 D1=4195290 D2=8514098 C0=0
C1=45543 C2=40489 C3=28041 C4=26448 C5=32801 C6=28264 C7=8
model=MS5803_14BA device=/dev/i2c-2 address=118
```

If desired, a specification string that includes the model of the sensor, the I<sup>2</sup>C port and the address on the bus can be included along with the read command. Only the MS5803-14BA sensor is currently supported, but others could be added in future versions:

```
$ TempPress -gm "MS5803-14BA:/dev/i2c-2:0x76"
18.75C 101.97kPA @2015-09-30 01:31:27.062553
$ TempPress -g --model "MS5803-14BA:/dev/i2c-99:0x76"
no-temperature no-pressure **FAILED** Failed to open the i2c device
(/dev/i2c-99) @2015-09-30 01:31:41.936991
$ TempPress -gm "MS5803-14BA:/dev/i2c-2:0x77"
no-temperature no-pressure **FAILED** Failed write @2015-09-30
01:31:52.952658
```

## Commands

TempPress supports the following commands from WGMS or via ServerExec:

### **tp --alarmdelay**

Returns the current setting for alarm delay, the number of seconds after announcing an alarm before another change in state can be announced.

```
$ ServerExec TempPress 'tp --alarmdelay'
600
```

**tp --alarmdelay** *delay-in-sec*

Sets a new value for alarm delay, the number of seconds after announcing an alarm before another change in state can be announced. This value is stored persistently in TempPress-persist-config.xml and reloaded when the server starts again.

```
$ ServerExec TempPress 'tp --alarmdelay 1200'
1200
```

**tp --autorun**

**tp --autostart**

Returns the value of the "auto run" Boolean flag (true or false), which determines whether sampling starts automatically when the program starts (autorun=true) or if the Start Sampling command must be issued to begin sampling (autorun=false)

```
$ ServerExec TempPress 'tp --autostart'
true
$ ServerExec TempPress 'tp --autorun'
true
```

**tp --autorun *new-value***

**tp --autostart *new-value***

Sets a new value of the "auto run" Boolean flag (true or false), which determines whether sampling starts automatically the program starts (autorun=true) or if the Start Sampling command must be issued to begin sampling (autorun=false) If the service is currently sampling, the response indicates not only the new setting but also warns the user that a separate stop command is required to stop sampling in present time (this setting only affects whether sampling automatically starts when the service starts up.) This value is stored persistently in TempPress-persist-config.xml and reloaded when the server starts again.

```
$ ServerExec TempPress 'tp --autorun false'
false (issue stop command to stop sampling)
$ ServerExec TempPress 'tp --autorun true'
true
```

**tp --clockaligned**

Returns the value of the "clock aligned" Boolean flag (true or false), which determines whether samples are taken according to the sampling interval aligned with the clock (clockaligned=true) or if the sampling interval simply starts when the service starts up (clockaligned=false) For example, if the sampling interval is 60 seconds, then aligning with the clock means that samples are taken at the top of every minute (12:00:00, 12:01:00, 12:02:00, etc.); or if the sampling interval is set to 1200 seconds (20 minutes), samples are taken at the top of the hour, 20 minutes and 40 minutes after the hour.

```
$ ServerExec TempPress 'tp --clockaligned'
true
```

**tp --clockaligned *new-value***

Sets the new value of the "clock aligned" Boolean flag (true or false), which determines whether samples are taken according to the sampling interval aligned with the clock (clockaligned=true) or if the sampling interval simply starts when the service starts up (clockaligned=false) If the setting is changed and the service is sampling, the result indicates not only the new value but also that sampling has restarted. This value is stored persistently in TempPress-persist-config.xml and reloaded when the server starts again.

```
$ ServerExec TempPress 'tp --clockaligned true'
true
$ ServerExec TempPress 'tp --clockaligned false'
false (restarted sampling)
```

**tp --getlast**

Returns the last reading from the sensor or "nil" if no reading has been taken.

```
$ ServerExec TempPress 'tp --getlast'
18.72C 101.91kPA @2015-09-30 02:39:00.000401
$ ServerExec TempPress 'tp --getlast'
no-temperature no-pressure **FAILED** Simulated error (simulated)
@2015-09-30 02:43:00.000370
$ ServerExec TempPress 'tp --getlast'
16.84C 112.46kPA (simulated) @2015-09-30 02:44:10.000376
```

**tp --log**

Returns the value of the "logging" Boolean flag (true or false), which determines whether samples, sensor information and calibration values are written to a logging CSV file (log=true) The default location for the log file (configured through TempPress-config.xml) is /mnt/XDATA/TempPress/TempPress.CSV. Logging is performed using the CSVSampleFileSet component. The first column of each line is the type of data, and the first appearance of a data type in the file indicates the headings for the data. The data types logged are:

- Calibration Data (CALIB)
- Sensor Readings (DATA)
- Sensor Information (INFO)
- Start of Sampling (START)
- End of Sampling (STOP)

For example:

```
CALIB,Date,Time,Millisec,Latitude,Longitude,C0,C1,C2,C3,C4,C5,C6,C7
DATA,Date,Time,Millisec,Latitude,Longitude,Temperature_C,Pressure_kPa,D
1,D2,Reported,Simulated,Success,ErrorMessage
INFO,Date,Time,Millisec,Latitude,Longitude,Model,Device,Address
START,Date,Time,Millisec,Latitude,Longitude
STOP,Date,Time,Millisec,Latitude,Longitude
START,2015/09/29,20:34:06,392,0,0
INFO,2015/09/29,20:35:00,000,37.4107432,-
122.0043553,MS5803_14BA,/dev/i2c-2,118
CALIB,2015/09/29,20:35:00,000,37.4107432,-
122.0043553,0,45543,40489,28041,26448,32801,28264,8
DATA,2015/09/29,20:35:00,000,37.4107432,-
122.0043553,18.83,102.07,4196862,8514086,false,false,true,
DATA,2015/09/29,20:36:00,000,37.410713,-
122.0043008,18.85,102.14,4197144,8514680,false,false,true,
DATA,2015/09/29,20:37:00,000,37.410778,-
122.0043883,18.80,102.04,4196736,8513350,false,false,true,
DATA,2015/09/29,20:38:00,000,37.4108712,-
122.004383,18.85,101.96,4196306,8514602,false,false,true,
DATA,2015/09/29,20:39:00,000,37.4107328,-
122.0045002,18.83,102.13,4197118,8514116,false,false,true,
DATA,2015/09/29,20:40:00,000,37.4108063,-
122.0043345,18.91,102.09,4196884,8516584,true,false,true,
```

```
$ ServerExec TempPress 'tp --log'  
true
```

**tp --log *new-value***

Sets the value of the "logging" Boolean flag (true or false), which determines whether samples, sensor information and calibration values are written to a logging CSV file (log=true) This value is stored persistently in TempPress-persist-config.xml and reloaded when the server starts again.

```
$ ServerExec TempPress 'tp --log true'  
true
```

**tp --nonreport**

Returns the value of the non-reporting sample interval, which determines spacing of sensor sampling (seconds) that is reported locally to other servers via CORBA, but is not reported via WGMS. If the value is set to 0, then the sensor is not sampled for local use.

```
$ ServerExec TempPress 'tp --nonreport'  
60
```

**tp --nonreport *new-value***

Sets the value of the non-reporting sample interval, which determines spacing of sensor sampling (seconds) that is reported locally to other servers via CORBA, but is not reported via WGMS. If the value is set to 0, then the sensor is not sampled for local use. If the setting is changed and the service is sampling, the result indicates not only the new value but also that sampling has restarted. This value is stored persistently in TempPress-persist-config.xml and reloaded when the server starts again.

```
$ ServerExec TempPress 'tp --nonreport 60'  
60  
$ ServerExec TempPress 'tp --nonreport 120'  
120 (restarted sampling)
```

**tp --pressurerange**

Returns the range of pressure readings (kPa) that is considered valid. Pressure readings that fall outside of the valid range generate low pressure or high pressure alarms.

```
$ ServerExec TempPress 'tp --pressurerange'  
105.00 to 130.00 kPa
```

**tp --pressurerange *lower-bound upper-bound***

Sets the range of pressure readings (kPa) that is considered valid. Pressure readings that fall outside of the valid range generate low pressure or high pressure alarms. This value is stored persistently in TempPress-persist-config.xml and reloaded when the server starts again.

```
$ ServerExec TempPress 'tp --pressurerange 105 130'  
105.00 to 130.00 kPa  
smc@dm3730_zoom_torpedo:~$ ServerExec TempPress 'tp --pressurerange  
104.5 130.5'  
104.50 to 130.50 kPa
```

**tp --report**

Returns the value of the reporting sample interval, which determines spacing of sensor sampling (seconds) that is reported as telemetry. If the value is set to 0, then sensor readings are not sampled for the purpose of reporting to WGMS, and no telemetry is sent.

```
$ ServerExec TempPress 'tp --report'
1200
```

#### **tp --report new-value**

Sets the value of the reporting sample interval, which determines spacing of sensor sampling (seconds) that is reported as telemetry. If the value is set to 0, then sensor readings are not sampled for the purpose of reporting to WGMS, and no telemetry is sent. If the setting is changed and the service is sampling, the result indicates not only the new value but also that sampling has restarted. This value is stored persistently in TempPress-persist-config.xml and reloaded when the server starts again.

```
$ ServerExec TempPress 'tp --report 1200'
1200
$ ServerExec TempPress 'tp --report 3600'
3600 (restarted sampling)
```

#### **tp --temprange**

Returns the range of temperature readings (C) that is considered valid. Temperature readings that fall outside of the valid range generate low temperature or high temperature alarms.

```
$ ServerExec TempPress 'tp --temprange'
1.00 to 45.00 C
```

#### **tp --temprange lower-bound upper-bound**

Sets the range of temperature readings (C) that is considered valid. Temperature readings that fall outside of the valid range generate low pressure or high pressure alarms. This value is stored persistently in TempPress-persist-config.xml and reloaded when the server starts again.

```
$ ServerExec TempPress 'tp --temprange 1 45'
1.00 to 45.00 C
$ ServerExec TempPress 'tp --temprange 0.35 45.65'
0.35 to 45.65 C
```

#### **tp --sim**

Returns the current simulation mode. The supported modes are:

- **OFF:** readings are taken from a live sensor
- **ERROR:** simulation of a sensor that has lost communication, reports NAN for readings and "Simulation error"; readings are marked as simulated
- **FIXED\_VALUE:** fixed user-settable values are reported for temperature and pressure and status marked as simulated
- **SINUSOID:** temperature and pressure values fluctuate over a 24-hour period according to a sine wave, peaking at noon UTC and hitting the lowest point at midnight UTC; readings are marked as simulated

```
$ ServerExec TempPress 'tp --sim'
OFF
$ ServerExec TempPress 'tp --sim'
```

```
ERROR
$ ServerExec TempPress 'tp --sim'
FIXED_VALUE (T=20.00C P=110.00kPa)
$ ServerExec TempPress 'tp --sim'
SINUSOID
```

**tp --sim off****tp --sim error****tp --sim fixed temp-value pressure-value****tp --sim fixed\_value temp-value pressure-value****tp --sim sinusoid**

Sets the simulation mode. If the setting is changed and the service is sampling, the result indicates not only the new value but also that sampling has restarted. This value is stored persistently in TempPress-persist-config.xml and reloaded when the server starts again.

```
$ ServerExec TempPress 'tp --sim off'
OFF
$ ServerExec TempPress 'tp --sim error'
ERROR (restarted sampling)
$ ServerExec TempPress 'tp --sim fixed 20 110'
FIXED_VALUE (T=20.00C P=110.00kPa) (restarted sampling)
$ ServerExec TempPress 'tp --sim sinusoid'
SINUSOID (restarted sampling)
```

**tp --start**

Instructs the service to start sampling according to configured sampling regimen. Note that when the "auto run" flag is enabled, the sensor automatically starts sampling when the service starts; but settings "auto run" on does not cause sampling to begin immediately.

```
$ ServerExec TempPress 'tp --start'
OK
```

**tp --status**

Returns the status of the service as a readable report, including:

- the last reading that was taken
- the state of the "auto run" and "clock aligned" flags
- the non-reporting and reporting sampling intervals
- the state of communication with the device (true=OK, false=not communicating)
- the state of sampling (true=sampling, false=not sampling)
- alarm states ("\*\* LOW-PRESSURE", "\*\* HIGH-PRESSURE", "\*\* LOW-TEMP", "\*\* HIGH-TEMP", "\*\* COMM-FAIL")
- simulation state

```
$ ServerExec TempPress 'tp --status'
AutoRun=true Comm=true ClockAligned=true GenInterval=60s
ReportInterval=1200s Sampling=true LastReading=18.91C 101.85kPa @2015-
09-30 03:36:00.000401 **LOW-PRESSURE
$ ServerExec TempPress 'tp --status'
AutoRun=true Comm=false ClockAligned=true GenInterval=10s
ReportInterval=1200s Simulate=ERROR Sampling=true LastReading=no-temper
ature no-pressure **FAILED** Simulated error (simulated) @2015-09-30 03
:37:30.000363
$ ServerExec TempPress 'tp --status'
```

```
AutoRun=true Comm=true ClockAligned=true GenInterval=10s ReportInterval=1200s Simulate=FIXED_VALUE (T=20.00C P=110.00kPa) Sampling=true LastReading=20.00C 110.00kPa (simulated) @2015-09-30 03:38:40.000508
```

### tp --stop

Instructs the service to stop sampling.

```
$ ServerExec TempPress 'tp --start'
OK
```

Normal user commands are displayed in WGMS under SV3 Commands as follows:

| Payload             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|---------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| G2_GPSListener      | <input type="button" value="Get Last Reading"/>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| G2_PayloadInterface | <input type="button" value="Get Logging"/>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <b>G2_TempPress</b> | <input type="button" value="Get Pressure Range"/><br><input type="button" value="Get Report Interval"/><br><input type="button" value="Get Temperature Range"/><br><input type="button" value="Set Logging"/> Enabled (true/false) <input type="text"/><br><input type="button" value="Set Pressure Range"/> Lower threshold (kPa) <input type="text"/> Upper threshold (kPa) <input type="text"/><br><input type="button" value="Set Report Interval"/> Interval (sec) <input type="text"/><br><input type="button" value="Set Temperature Range"/> Lower threshold (C) <input type="text"/> Upper threshold (C) <input type="text"/><br><input type="button" value="Start Sampling"/><br><input type="button" value="Status"/><br><input type="button" value="Stop Sampling"/> |

When expert mode is enabled, the display is as follows:

| Payload             |                                |                                                                                       |
|---------------------|--------------------------------|---------------------------------------------------------------------------------------|
| G2_GPSListener      | Get Alarm Delay                | ⚠                                                                                     |
| G2_PayloadInterface | Get Auto Run                   | ⚠                                                                                     |
| G2_TempPress        | Get Clock Aligned              | ⚠                                                                                     |
|                     | Get Last Reading               |                                                                                       |
|                     | Get Logging                    |                                                                                       |
|                     | Get Non-Report Interval        | ⚠                                                                                     |
|                     | Get Pressure Range             |                                                                                       |
|                     | Get Report Interval            |                                                                                       |
|                     | Get Temperature Range          |                                                                                       |
|                     | Set Alarm Delay                | ⚠ Time (sec) <input type="text"/>                                                     |
|                     | Set Auto Run                   | ⚠ Enabled (true/false) <input type="text"/>                                           |
|                     | Set Clock Aligned              | ⚠ Clock-aligned (true/false) <input type="text"/>                                     |
|                     | Set Logging                    | Enabled (true/false) <input type="text"/>                                             |
|                     | Set Non-Report Interval        | ⚠ Interval (sec) <input type="text"/>                                                 |
|                     | Set Pressure Range             | Lower threshold (kPa) <input type="text"/> Upper threshold (kPa) <input type="text"/> |
|                     | Set Report Interval            | Interval (sec) <input type="text"/>                                                   |
|                     | Set Temperature Range          | Lower threshold (C) <input type="text"/> Upper threshold (C) <input type="text"/>     |
|                     | Start Sampling                 |                                                                                       |
|                     | Status                         |                                                                                       |
|                     | Stop Sampling                  |                                                                                       |
|                     | Turn Error Simulation On       | ⚠                                                                                     |
|                     | Turn Fixed Value Simulation On | ⚠ Temperature (C) <input type="text"/> Pressure (kPa) <input type="text"/>            |
|                     | Turn Simulation Off            | ⚠                                                                                     |
|                     | Turn Sinusoidal Simulation On  | ⚠                                                                                     |

## Alarms

"Communication to sensor failed" is sent whenever there is an error reading or writing values to the sensor; this is noted in the status as "\*\*\* COMM-FAIL"

"Pressure exceeds maximum threshold" is sent whenever the pressure reading is above the configured maximum (see "tp --pressurerange"); this is noted in the status as "\*\*\* HIGH-PRESSURE"

"Pressure below minimum threshold" is sent whenever the pressure reading is below the configured minimum (see "tp --pressurerange"); this is noted in the status as "\*\*\* LOW-PRESSURE"

"Temperature exceeds maximum threshold" is sent whenever the temperature reading is above the configured maximum (see "tp --temprange"); this is noted in the status as "\*\*\* HIGH-TEMP"

"Temperature below minimum threshold" is sent whenever the temperature reading is below the configured minimum (see "tp --temprange"); this is noted in the status as "\*\*\* LOW-TEMP"

### CORBA Events

Samples are reported via CORBA with the TempPressSample event name and the following name/value pairs (in addition to those supported by TelemetryEventBase):

- "TEMPPRESS\_PRESSURE\_KPA" is a double-precision floating number representing the pressure (kPa); NAN indicates that there was an error that prevented a reading from being taken
- "TEMPPRESS\_TEMPERATURE\_C" is a double-precision floating point number representing the temperature (c); NAN indicates that there was an error that prevented a reading from being taken
- "TEMPPRESS\_PRESSURE\_KPA\_SINGLE" is a single-precision floating number representing the pressure (kPa); NAN indicates that there was an error that prevented a reading from being taken
- "TEMPPRESS\_TEMPERATURE\_C\_SINGLE" is a single-precision floating point number representing the temperature (c); NAN indicates that there was an error that prevented a reading from being taken
- "TEMPPRESS\_D1" is 32-bit unsigned integer with the D1 raw reading from the sensor; it is the primary basis for the pressure reading in combination with the D2 value and calibration constants
- "TEMPPRESS\_D2" is 32-bit unsigned integer with the D2 raw reading from the sensor; it is the primary basis for the temperature reading in combination calibration constants
- "TEMPPRESS\_SIMULATED" is a Boolean value that is true if the reading was simulated or false if the reading was from the sensor
- "TEMPPRESS\_SUCCESS" is a Boolean value that is true if the reading was successful or false if the readings are invalid because an error occurred in the acquisition process
- "TEMPPRESS\_FLAG0" is an 8-bit unsigned integer with the 1's bit equal to the "simulated" flag and the 2's bit equal to the reverse of the "success" flag: a value of zero means that the reading is good and not simulated
- "TEMPPRESS\_ERRORMESSAGE" is a string containing any error message that might have been generated (blank if successful)
- "TEMPPRESS\_MODEL" is a string containing the model name for the sensor (for example, "MS5803\_14BA")
- "TEMPPRESS\_ADDRESS" is a 16-bit unsigned integer with the I2C address of the sensor on whatever bus it is attached to
- "TEMPPRESS\_DEVICE" is a string containing the device used to access to the I2C sensor (for example, "/dev/i2c-2")
- "TEMPPRESS\_NEWCAL" is a Boolean value that is true if this sample contains calibration constants that were obtained for the first time with this sample
- "TEMPPRESS\_C0" through "TEMPPRESS\_C7" are 16-bit unsigned integers with calibration constants read from the sensor

**Telemetry Sub-Type 0 (deprecated)**

PayloadInterface-telemetry-config.xml contains the following entry to convert TempPressSample events to type 0x91 telemetry type ID 159 and subtype 0:

```
<Element>
  <first boardID="0xFF" taskID="0x32" />
  <second>
    <key boardID="0xFF" taskID="0x32" />
    <payloadType value="0x0000009F" />
    <eventNameToFormatCode>
      <Element>
        <first value="TempPressSample" />
        <second value="0" />
      </Element>
    </eventNameToFormatCode>
    <reportParams>
      <Element>
        <first value="0" />
        <second maxSamples="1" targetAddress="2">
          <reportHeaderFormat>
            <fields>
              <Element libField="Header91" />
              <Element libField="Raw" eventField="Telem-LengthToFollow" />
              <Element libField="Raw" eventField="Telem-PayloadType" />
              <Element libField="Raw" eventField="Telem-BoardID" />
              <Element libField="Raw" eventField="Telem-TaskID" />
              <Element libField="FC0BinaryNormal" />
              <Element libField="RawINT8" eventField="Telem-NumSamples" />
            </fields>
          </reportHeaderFormat>
          <sampleHeaderFormat>
            <fields>
              <Element libField="Raw" eventField="Telem-Latitude" />
              <Element libField="Raw" eventField="Telem-Longitude" />
              <Element libField="Raw" eventField="Telem-TimeSec" />
              <Element libField="Raw"
eventField="TEMPPRESS_TEMPERATURE_C_SINGLE" />
              <Element libField="Raw"
eventField="TEMPPRESS_PRESSURE_KPA_SINGLE" />
              <Element libField="Raw" eventField="TEMPPRESS_FLAG0" />
              <Element libField="Raw"
eventField="TEMPPRESS_ERRORMESSAGE" />
            </fields>
          </sampleHeaderFormat>
        </second>
      </Element>
    </reportParams>
  </second>
</Element>
```

The reported telemetry in WGMS looks like this:

| TimeStamp           | Vehicle | WGMS Tl... | Board Id | Task Id | Latitude(deg) | Longitude(deg) | Temperature (C) | Pressure (kPa) | Simulated | Error | Error Message |
|---------------------|---------|------------|----------|---------|---------------|----------------|-----------------|----------------|-----------|-------|---------------|
| 09/30/2015 19:40:00 | Abalone | 09/3...    | 33       | 50      | 37.41073      | -122.00442     | 18.6700         | 101.9900       | 0         | 0     |               |
| 09/30/2015 19:40:00 | Abalone | 09/3...    | 32       | 50      | 37.41073      | -122.00442     | 18.6800         | 102.5900       | 0         | 0     |               |
| 09/30/2015 19:20:00 | Abalone | 09/3...    | 33       | 50      | 37.41084      | -122.00444     | 18.5800         | 101.7700       | 0         | 0     |               |
| 09/30/2015 19:20:00 | Abalone | 09/3...    | 32       | 50      | 37.41084      | -122.00444     | 18.5900         | 102.5500       | 0         | 0     |               |
| 09/30/2015 19:00:00 | Abalone | 09/3...    | 33       | 50      | 37.41086      | -122.00442     | 18.5500         | 102.0800       | 0         | 0     |               |
| 09/30/2015 19:00:00 | Abalone | 09/3...    | 32       | 50      | 37.41086      | -122.00442     | 18.5900         | 102.5500       | 0         | 0     |               |
| 09/30/2015 18:40:00 | Abalone | 09/3...    | 33       | 50      | 37.41075      | -122.00469     | 18.6100         | 101.9000       | 0         | 0     |               |
| 09/30/2015 18:40:00 | Abalone | 09/3...    | 32       | 50      | 37.41075      | -122.00469     | 18.6100         | 102.5600       | 0         | 0     |               |
| 09/30/2015 18:20:00 | Abalone | 09/3...    | 33       | 50      | 37.41059      | -122.00459     | 18.7000         | 102.0100       | 0         | 0     |               |
| 09/30/2015 18:20:00 | Abalone | 09/3...    | 32       | 50      | 37.41059      | -122.00459     | 18.6300         | 102.5500       | 0         | 0     |               |
| 09/30/2015 18:00:00 | Abalone | 09/3...    | 33       | 50      | 37.41081      | -122.00437     | 18.5200         | 102.1900       | 0         | 0     |               |
| 09/30/2015 18:00:00 | Abalone | 09/3...    | 32       | 50      | 37.41081      | -122.00437     | 18.6000         | 102.6000       | 0         | 0     |               |
| 09/30/2015 17:40:00 | Abalone | 09/3...    | 33       | 50      | 37.41095      | -122.00449     | 18.5700         | 102.2100       | 0         | 0     |               |
| 09/30/2015 17:40:00 | Abalone | 09/3...    | 32       | 50      | 37.41095      | -122.00449     | 18.5800         | 102.6100       | 0         | 0     |               |
| 09/30/2015 17:20:00 | Abalone | 09/3...    | 33       | 50      | 37.41102      | -122.00466     | 18.5400         | 102.0500       | 0         | 0     |               |
| 09/30/2015 17:20:00 | Abalone | 09/3...    | 32       | 50      | 37.41102      | -122.00466     | 18.5100         | 102.6800       | 0         | 0     |               |
| 09/30/2015 17:00:00 | Abalone | 09/3...    | 33       | 50      | 37.41082      | -122.00446     | 18.3400         | 102.0600       | 0         | 0     |               |
| 09/30/2015 17:00:00 | Abalone | 09/3...    | 32       | 50      | 37.41082      | -122.00446     | 18.3900         | 102.6700       | 0         | 0     |               |
| 09/30/2015 16:40:00 | Abalone | 09/3...    | 33       | 50      | 37.41086      | -122.00434     | 18.5800         | 102.0300       | 0         | 0     |               |
| 09/30/2015 16:40:00 | Abalone | 09/3...    | 32       | 50      | 37.41086      | -122.00434     | 18.5200         | 102.6600       | 0         | 0     |               |
| 09/30/2015 16:20:00 | Abalone | 09/3...    | 33       | 50      | 37.41073      | -122.00448     | 18.6400         | 102.1500       | 0         | 0     |               |
| 09/30/2015 16:20:00 | Abalone | 09/3...    | 32       | 50      | 37.41073      | -122.00448     | 18.5000         | 102.6900       | 0         | 0     |               |

The WGMS column XML for this view looks like this:

```
<columns>
<column type="id">PayloadPacketRecordId</column>
<column type="bound" width="160px" sort="0">TimeStamp</column>
<column type="related" entitytype="48" width="120px">VehicleId</column>
<column type="related" entitytype="37" width="80px" name="WGMS
TimeStamp">PayloadPacketId</column>
<column type="related" width="80px" entitytype="37" basecolumn="GliderBoardAddress"
filter="1">GliderBoardAddress</column>
<column type="related" width="80px" entitytype="37" basecolumn="GliderTaskAddress"
filter="1">GliderTaskAddress</column>
<column type="bound" width="120px">Latitude</column>
<column type="bound" width="120px">Longitude</column>
<column type="calculated" width="120px" method="bytefield|doublesigned|0|4"
name="Temperature (C)">RecordData</column>
<column type="calculated" width="120px" method="bytefield|doublesigned|4|4"
name="Pressure (kPa)">RecordData</column>
<column type="calculated" width="80px"
method="expression=if(IEEERemainder([%bytefield|int|8|1%],2),1,0)"
name="Simulated">RecordData</column>
<column type="calculated" width="80px"
method="expression=if([%bytefield|int|8|1%]>1,1,0)" name="Error">RecordData</column>
<column type="calculated" width="540px" method="bytefield|string|9|-1" name="Error
Message">RecordData</column>
</columns>
```

### Telemetry Sub-Type 1

Telemetry sub-type 1 was added to allow AMPS modules to report temperature and pressure as telemetry, as well. The following fields were added to the end of the message, just before the error string; if the message is shortened not to include them, it is assumed that the device is not related to AMPS:

- AMPS power domain number (0=no AMPS domain)
- PDC serial number tail

## Format code 1 (AMPS sensor compatible)

| Field Name             | Bytes Used | Byte #       | Format                                                                                   |
|------------------------|------------|--------------|------------------------------------------------------------------------------------------|
| <u>PayloadType</u>     | 4          | 0-3          | 159 (Temperature and Pressure)                                                           |
| Board Id               | 1          | 4            | Uin8                                                                                     |
| Task Id                | 1          | 5            | Uin8                                                                                     |
| Sensor Data Format     | 1          | 6 (bits 0-3) | Unsigned – 1                                                                             |
| Default Parser         | 1          | 6 (bit 4)    | 0=Binary/1=ASCII – 0                                                                     |
| Delivery Priority      | 1          | 6 (bits 5-7) | Unsigned – 0                                                                             |
| Repeat Count           | 1          | 7            | Uin8                                                                                     |
| Latitude               | 4          | 8-11         | Int32, degrees*600,000                                                                   |
| Longitude              | 4          | 12-15        | Int32, degrees*600,000                                                                   |
| Time                   | 4          | 16-19        | Unix <u>time_t</u> as Int32                                                              |
| Temperature            | 4          | 20-23        | Single-precision floating point, degrees C                                               |
| Pressure               | 4          | 24-27        | Single-precision floating point, <u>kPa</u>                                              |
| Simulated              | 1          | 28 (bit 0)   | 1=simulated reading, 0=sensor reading                                                    |
| Error                  | 1          | 28 (bit 1)   | 1=error, 0=success                                                                       |
| AMPS Power Domain      | 1          | 29           | 0=Not AMPS device, 1-255=AMPS domain number                                              |
| PDC Serial Number Tail | 4          | 30-33        | The last 4 bytes of the PDC serial number as a string (spaces or missing=not AMPS based) |
| Error Message          | Variable   | 34-          | Free form error message for this reading (blank or missing if success)                   |

The entry in PayloadInterface-telemetry-config.xml was modified to be:

```

<Element>
  <first boardID="0xFF" taskID="0x32" />
  <second>
    <key boardID="0xFF" taskID="0x32" />
    <payloadType value="0x0000009F" />
    <eventNameToFormatCode>
      <Element>
        <first value="TempPressSample" />
        <second value="1" />
      </Element>
    </eventNameToFormatCode>
    <reportParams>
      <Element>
        <first value="1" />
        <second maxSamples="1" targetAddress="2">
          <reportHeaderFormat>
            <fields>
              <Element libField="Header91" />
            </fields>
          </reportHeaderFormat>
        </second>
      </Element>
    </reportParams>
  </second>
</Element>

```

```

        <Element libField="Raw" eventField="Telem-LengthToFollow"
/>
        <Element libField="Raw" eventField="Telem-PayloadType" />
        <Element libField="Raw" eventField="Telem-BoardID" />
        <Element libField="Raw" eventField="Telem-TaskID" />
        <Element libField="FC1BinaryNormal" />
        <Element libField="RawINT8" eventField="Telem-NumSamples"
/>
    </fields>
</reportHeaderFormat>
<sampleHeaderFormat>
    <fields>
        <Element libField="Raw" eventField="Telem-Latitude" />
        <Element libField="Raw" eventField="Telem-Longitude" />
        <Element libField="Raw" eventField="Telem-TimeSec" />
        <Element libField="Raw"
eventField="TEMPPRESS_TEMPERATURE_C_SINGLE" />
        <Element libField="Raw"
eventField="TEMPPRESS_PRESSURE_KPA_SINGLE" />
        <Element libField="Raw" eventField="TEMPPRESS_FLAG0" />
        <Element libField="AMPSDomain0" />
        <Element libField="PDCSNBlank" />
        <Element libField="Raw"
eventField="TEMPPRESS_ERRORMESSAGE" />
    </fields>
</sampleHeaderFormat>
</second>
</Element>
</reportParams>
</second>
</Element>

```

The following elements were added to PayloadInterface-telemetry-config.xml's field library section:

```

<Element>
    <key value="AMPSDomain0" />
    <value content="CA_LITERAL">
        <literal type="UINT8" value="0x00" />
    </value>
</Element>
<Element>
    <key value="PDCSNBlank" />
    <value content="CA_LITERAL">
        <literal type="STRING" value=" " />
    </value>
</Element>

```

The WGMS parser column XML was modified as follows for payload type ID 159 form code 1:

```

<columns>
<column type="id">PayloadPacketRecordId</column>
<column type="bound" width="160px" sort="0">TimeStamp</column>
<column type="related" entitytype="48" width="120px">VehicleId</column>
<column type="related" entitytype="37" width="80px" name="WGMS
TimeStamp">PayloadPacketId</column>
<column type="related" width="80px" entitytype="37"
basecolumn="GliderBoardAddress" filter="1">GliderBoardAddress</column>
<column type="related" width="80px" entitytype="37"
basecolumn="GliderTaskAddress" filter="1">GliderTaskAddress</column>
<column type="bound" width="120px">Latitude</column>

```

```

<column type="bound" width="120px">Longitude</column>
<column type="calculated" width="120px" method="bytefield|doublesigned|0|4"
name="Temperature (C)">RecordData</column>
<column type="calculated" width="120px" method="bytefield|doublesigned|4|4"
name="Pressure (kPa)">RecordData</column>
<column type="calculated" width="80px"
method="expression=if(IEEERemainder([%bytefield|int|8|1%],2),1,0)"
name="Simulated">RecordData</column>
<column type="calculated" width="80px"
method="expression=if([%bytefield|int|8|1%]>1,1,0)"
name="Error">RecordData</column>
<column type="calculated" width="120px" method="bytefield|int|9|1" name="Power
Domain">RecordData</column>
<column type="calculated" width="120px" method="bytefield|string|10|4"
name="PDC S/N Tail">RecordData</column>
<column type="calculated" width="540px" method="bytefield|string|14|-1"
name="Error Message">RecordData</column>
</columns>

```

### Other Things to Note

- It is problematic to run a service as root in order to get permission to open a device, so the udev service configuration was modified to allow anyone to open I2C devices by adding /etc/udev/rules.d/99-i2c.rules with the following content:

```
SUBSYSTEM=="i2c-dev", MODE="0666"
```

This setting only takes effect after a system reboot (normally called for during the installation process before anything is run), but the following commands executed as root will cause udev to restart and re-read the configuration without a restart:

```
$ modprobe i2c-dev
$ udevadm trigger
```

See also <http://blog.chrysocome.net/2012/11/raspberry-pi-i2c.html>

- The shipping configuration only saves a small amount of data in the CSV logs (about 10Mb, which is the default setting for CSVFileSet): you may want to make changes in the configuration to retain more of the data or if you change the polling rate. The following is the shipping configuration in TempPress-config.xml:

```

<fileSetSettings class_id="13" tracking_level="0" version="0">
  <archiveCompressionType>NONE</archiveCompressionType>
  <archiveDirectory />
  <extension>.CSV</extension>
  <keepWritesTogether>1</keepWritesTogether>
  <maxBytesQueued>10000</maxBytesQueued>
  <maxFileSize>1000000</maxFileSize>
  <maxFiles>10</maxFiles>
  <maxRetainTime>00:00:00</maxRetainTime>
  <prefix>TempPress</prefix>
  <purgeStrategy>OLDEST</purgeStrategy>
  <workingDirectory>/mnt/XDATA/TempPress</workingDirectory>
</fileSetSettings>

```

The "maxFileSize" setting is the maximum number of bytes to store in a single file, while "maxFiles" is the maximum number of those files to keep. "maxFiles" can be modified to increase or decrease data retention. Keep in mind that the data log lines are typically about 100 bytes, and at one sample per minute the default settings will retain data for about 69 days. If there are a lot of errors, more space will be consumed because the error message will be included.

After changing TempPress-config.xml, the new settings will not take effect until the software is restarted.

- Calculation of the temperature and pressure based on D1, D2 and the calibration constants is covered in a document provided on the Measurement Specialties at <http://www.meas-spec.com/downloads/MS5803-14BA.pdf>

## timelimit2

The standard **timelimit** utility limits the clock runtime of a process to a given number of seconds; for example:

```
exec timelimit -pq -s 9 -t 600 scp $IDENT -P 10501  
vehicle@12.156.66.48:$DIRPART/runme workspace/runme
```

In this case, the "scp" command is limited to 600 seconds, and then is killed if it is not complete yet. Some issues with this were found, though, when the command creates a child task. In this configuration, timelimit only kills the top level process: the children continue to run.

To resolve this, **timelimit2** uses some classes already in use in the SMC framework classes to recursively kill the children as well as the parent. The syntax is as follows:

**timelimit2** -timeout=*ttt* [--signal=*sss*] [--debug] ...*command-line*...

The timeout value (*ttt*) is the maximum number of seconds to allow the process to run; the optional signal value (*sss*) is the signal number to use when killing processes, which defaults to 9 (SIGKILL)

## VemcoVR2C Configuration

The VemcoVR2C application is a module capable of interfacing to, initializing, and receiving data, over a serial port, from a Vemco VR2C sensor. All data received from the sensor is logged in the `data` file located at `/home/smc/XDATA/VR2C/vr2c.txt`. The VemcoVR2C application logs diagnostic messages, including communications with the sensor, in the log file `~/log/VemcoVR2C.log`. All Tag detections reported by the sensor (read from the serial port as ASCII strings) are sent as telemetry events **VR2CTagReport**, and status messages received from the sensor are periodically sent as telemetry events **VR2CStatusReport**.

The VemcoVR2C application is started from:

**~/roots/200\_VemcoVR2C/**

and is configured by the XML file:

**~/roots/200\_VemcoVR2C/VemcoVR2C-config.xml**

The VemcoVR2C instance is configured with the serverName **VR2C**, and assigned the taskId 21 (hexadecimal 15).

### Selected XML file keys

- **outputFilename** specifies the fully qualified name of the file where all data received from the sensor is recorded. Default setting is **/home/smc/XDATA/vr2c.txt**.

- **DefaultStatusReportSeconds** specifies the delay between the telemetry events VR2CStatusReport, based on the most recent status message from the sensor. Default setting is **3600 (1 hour)**.
- **serialPortSettings** for VemcoVR2C, the default serial port is /dev/ttyLX4. By default it is set to communicate with a baudrate of 9600, parity NONE, 1 stopbit, characterSize 8, and flowControl NONE. Ensure that the serial connector cables are plugged into the specified serial port on the SMC.
- **powerOnVerb** and **powerOffVerb** for VemcoVR2C, the command string references power switch. **TagPrefix** an ASCII string depicting the start of a valid tag detection field in the message received from the sensor. Default setting is **A69-**.
- **StatusIdentifier** an ASCII string identifier for the RTM status message received from the sensor. Default setting is **STS**.
- **ReportDelaySeconds** time, in seconds, for the gap/delay between successive telemetry events of tag detection samples. Default setting is **120**.
- 

### Using the VemcoVR2C

The VemcoVR2C application is controlled by the following commands:

"start"

powers on the sensor, attempts discovery of the sensor and if successful uses the discovered serial number for further comms with the sensor, initializes and starts the sensor for reporting tag detections and status (as RTM messages, ref: Vemco VR2C user manual). Upon completion, the response to this command will be the VR2C response to the INFO command or an error message if start was unsuccessful.

"stop"

sends stop command to the sensor, followed by a power off of the sensor.

"autostart [value]"

specifies whether to automatically start the sensor at subsequent SMC software startup. The response to this command returns the current state of the autostart feature, which can be queried by issuing this command without specifying a value (true, false).

"serialnumber [number]"

Attempts to use the specified serialnumber to communicate with the VR2C and if successful saves that as the default serialnumber to be used if discovery at start fails. Response to this command is:

'serialnumber in use' vr2c-info-response

If the attempted serialnumber fails, the 'serialnumber in use' will be the existing serial number in use prior to this attempt. If no serialnumber is specified, a response to the command is generated with the existing serial number in use and the VR2C's response to the INFO command.

"status [seconds]"

This will set the status report (status telemetry event) period in seconds (if specified), and return a response containing the existing/updated report period in seconds followed by the VR2C's response to the STATUS command: 'seconds' vr2c-status-response

SV3 Commands labels:

**Start**

**Stop**

**Get Info** (this uses the 'serialnumber' command with no parameter)

**Get Status** (this uses the 'status' command with no parameter)

**Get AutoStart** (this uses the 'autostart' command with no parameter)

**Set AutoStart <value>**

**Get SerialNumber** (this uses the 'serialnumber' command with no parameter)

**Set SerialNumber <value>** (this command is only available in expert mode)

## VirtualRelay Configuration

The VirtualRelay application is a module that uses the Virtual Relay protocol to communicate with WGMS over a TCP/IP connection. Samples methods of transports include WiFi, Ethernet, or a cell modem. These communication transports are managed by ConnectionManager.

VirtualRelay is dependent on ConnectionManager. VirtualRelay is also dependent on the FCC firmware, which will be in a future release (version 2.0.973 or newer).

The VirtualRelay application is started from

**~/roots/200\_VirtualRelay/**

And is configured by the XML file:

**/roots/200\_VirtualRelay/VirtualRelay-config.xml**

VirtualRelay is assigned the board ID 32 (0x20) and the task ID 170 (0xAA). Each channel is assigned its own task ID as well. For example, if a cell modem is configured, it should take task ID 171 (0xAB).

### Common XML file keys to edit

A typical configuration for the controller settings is as follows:

```
<controllerSettings>
  <count>1</count>
  <item_version>0</item_version>
  <item>
    <host>test.liquidr.com</host>
    <hostPort>643</hostPort>
    <commsTaskID>171</commsTaskID>
  </item>
</controllerSettings>
```

```
        <sleepTime>180</sleepTime>
        <retries>10</retries>
        <retryRate>20</retryRate>
        <timeoutSeconds>5</timeoutSeconds>
        <messageManagerSettings>
            <telemetryInterrupt>1</telemetryInterrupt>
            <ackInterrupt>1</ackInterrupt>
            <telemetryTimeoutCount>4</telemetryTimeoutCount>
            <protocolRevision>32</protocolRevision>
            <backupFilename>savedMessages-
test32.xml</backupFilename>
        </messageManagerSettings>
    </item>
</controllerSettings>
```

- **count:** This is the number of channels that the Virtual Relay will use.
- **host:** This is where the Virtual Relay will try to connect to send and receive messages. Typical values are test.liquidr.com (the test server) or packets.wgms.com (the production server).
- **hostPort:** The port that Virtual Relay uses to connect to the host. This is 643 for the test server and 443 for the production server.
- **commsTaskID:** This is the task ID that this particular channel uses to communicate with the FCC. Note that this should be different than the Virtual Relay's task ID of 170 (0xAA) and unique for every channel.
- **sleepTime:** This is the amount of time, in seconds, that the Virtual Relay will wait after the last communication with the host before starting to attempt a new session. Use this parameter to control how the channel is duty-cycled. A setting of 0 will disable duty cycling.
- **retries:** This is the number of times the Virtual Relay will try to connect to the host before giving up and waiting the amount of time specified by *sleepTime*.
- **retryRate:** This is the amount of time, in seconds, that the Virtual Relay will wait between attempted connections to the host. Make sure to set *retries* and *retryRate* such that the channel being used has enough time to be initialized (for example, if a cell modem takes 60 seconds to start up, make sure *retries\*retryRate* is greater than 60).
- **timeoutSeconds:** This is the timeout for the socket connection made between the Virtual Relay and the host. This should be set to five seconds at a minimum, but it can be set higher if there is latency in the connection (eg, a non-wired connection).
- **telemetryInterrupt:** This specifies whether or not a telemetry message will interrupt the channel if it is in sleep mode. (0=off, 1=on).
- **ackInterrupt:** This specifies whether or not an ack will interrupt the channel if it is in sleep mode. (0=off, 1=on).

- **telemetryTimeoutCount:** If the Virtual Relay accumulates this many telemetry messages without being able to communicate to the host, it will tell the FCC to default back to Iridium.
- **protocolRevision:** This is the protocolRevision that this channel uses to communicate. This determines which channel ConnectionManager is asked to turn on/off and the channel that this Virtual Relay reports as to the host.
- **backupFilename:** In the event that the SMC is powered off while there are messages waiting to be sent back to the host, they will be stored in this file and read when the Virtual Relay next starts up.

### Typical Virtual Relay configurations

| Setting          | WiFi/Ethernet    |                  | Cell Modem       |                  |
|------------------|------------------|------------------|------------------|------------------|
|                  | Production       | Test             | Production       | Test             |
| <b>Host</b>      | packets.wgms.com | test.liquidr.com | packets.wgms.com | test.liquidr.com |
| hostPort         | 443              | 643              | 443              | 643              |
| sleepTime        | 180              | 180              | 200              | 200              |
| retries          | 10               | 10               | 15               | 15               |
| retryRate        | 20               | 20               | 20               | 20               |
| timeoutSeconds   | 5                | 5                | 15               | 15               |
| protocolRevision | 32               | 32               | 33               | 33               |

### Commands

A user can change some of the Virtual Relay settings through commands sent to the process VirtualRelay via ServerExec, or to the Virtual Relay task (task ID 170, ox 0xAA). These settings will be saved in the xml configuration file and will persist after a reboot.

- `setsleeptime channelTaskID newSleepTime`  
Changes the sleepTime configuration for the channel specified by the task ID. For example, a sample command may be:

```
setsleeptime 171 60
```

- `setretryrate channelTaskID newChannelTaskID`  
Changes the retryRate configuration for the channel specified by the task ID. For example, a sample command may be:

```
setretryrate 171 15
```

- `setnumretries channelTaskID newNumRetries`  
Changes the retries configuration for the channel specified by the task ID. For example, a sample command may be:

setnumretries 171 5

- `setteleminterrupt channelTaskID newTelemInterrupt`  
Changes the telemetryInterrupt configuration for the channel specified by the task ID. For example, a sample command may be:

setteleminterrupt 171 0

- `setackinterrupt channelTaskID newAckInterrupt`  
Changes the ackInterrupt configuration for the channel specified by the task ID. For example, a sample command may be:

setackinterrupt 171 0

- `settimeout channelTaskID newTimeout`  
Changes the timeoutSeconds configuration for the channel specified by the task ID. For example, a sample command may be:

settimeout 171 10

## WetLabsSensorSimulator

The utility `~/bin/WetLabsSensorSimulator` is designed to generate simulated Eco Puck samples, provide configuration and menu responses to sensor commands, and emulate the sensor's start and stop states when responding to these commands. The commands are received and simulated responses sent over a pseudo serial port, specified when starting the utility.

In order use the simulated communications, the WetLabsSensor application must be configured for communicating with an Eco Puck sensor (configured in `~/roots/200_EcoSensor`) and using the pseudo serial port specified when invoking the WetLabsSensorSimulator.

Example:

```
WetLabsSensorSimulator /home/smc/roots/200_EcoSensor/pseudo
```

## Appendix 1 - Sample Application Configuration File

The following is a sample application configuration file listing taken from the GPS Listener process (GPSListener-config.xml):

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE boost_serialization>
<boost_serialization signature="serialization::archive" version="10">
  <rootlevel class_id="0" tracking_level="0" version="7">
    <loggerSettings class_id="1" tracking_level="0" version="0">
      <logLevel>LT_WARNING</logLevel>
      <processName>GPSListener</processName>
      <isFileDes>1</isFileDes>
      <fileDes>1</fileDes>
      <fileName/>
      <pipeName/>
    </loggerSettings>
    <serialPortSettings class_id="2" tracking_level="0" version="0">
      <portName>/dev/ttyO1</portName>
      <baudRate>4800</baudRate>
      <parity>NONE</parity>
      <characterSize>8</characterSize>
      <stopBits>1</stopBits>
      <flowControl>NONE</flowControl>
      <rts>1</rts>
      <dtr>1</dtr>
      <breakTimeoutMultiplier>1</breakTimeoutMultiplier>
    </serialPortSettings>
    <maxString>1000</maxString>
    <readTimeoutMs>50</readTimeoutMs>
    <serverName>GPSListener</serverName>
    <serverKind/>
    <dateSetDirectly>1</dateSetDirectly>
    <dateVerb>sudo date --utc {mm}{dd}{HH}{MM}{yyyy}.{SS}</dateVerb>
    <hwClockVerb>sudo hwclock -w</hwClockVerb>
    <defaultOutputFileName>/mnt/XDATA/GPS/sensor.out</defaultOutputFileName>
    <deltaToSetTimeMs>100</deltaToSetTimeMs>
    <deltaToSetHWClockMs>0</deltaToSetHWClockMs>
    <rmsAccuracyThreshold>50</rmsAccuracyThreshold>
    <gpsEventSpecs class_id="3" tracking_level="0" version="0">
      <count>2</count>
      <item_version>1</item_version>
      <item class_id="4" tracking_level="0" version="1">
        <divisor>5</divisor>
        <divisorMode>0</divisorMode>
        <eventSettings class_id="5" tracking_level="0" version="0">
          <eventName>GPSSample</eventName>
          <eventType class_id="6" tracking_level="0" version="0">
            <typeName>GPSSample</typeName>
            <domainName>LiquidR-SMC</domainName>
          </eventType>
        </eventSettings>
      </item>
      <item>
        <divisor>1</divisor>
        <divisorMode>0</divisorMode>
        <eventSettings>
          <eventName>GPSSample1Sec</eventName>
          <eventType>
            <typeName>GPSSample</typeName>
            <domainName>LiquidR-SMC</domainName>
          </eventType>
        </eventSettings>
      </item>
    </gpsEventSpecs>
    <tryBaudRates>
      <count>2</count>
    </tryBaudRates>
  </rootlevel>
</boost_serialization>
```

```

    <item_version>0</item_version>
    <item>4800</item>
    <item>9600</item>
  </tryBaudRates>
  <setTimeOnPPS>true</setTimeOnPPS>
  <ppsGPIO>167</ppsGPIO>
  <systemGPSOffline>
    <eventName>SystemGPSOffline</eventName>
    <eventType>
      <typeName>EventBase</typeName>
      <domainName>LiquidR-SMC</domainName>
    </eventType>
  </systemGPSOffline>
  <systemGPSOnline>
    <eventName>SystemGPSOnline</eventName>
    <eventType>
      <typeName>EventBase</typeName>
      <domainName>LiquidR-SMC</domainName>
    </eventType>
  </systemGPSOnline>
  <hostGPSEventSettings>
    <eventName>HostGPSSample</eventName>
    <eventType>
      <typeName>GPSSample</typeName>
      <domainName>LiquidR-SMC</domainName>
    </eventType>
  </hostGPSEventSettings>
  <eventTransceiverSettings class_id="8" tracking_level="0" version="2">
    <eventServiceName>EventManager</eventServiceName>
    <eventServiceKind/>
    <receiveEnable>1</receiveEnable>
    <aliasToEvent class_id="9" tracking_level="0" version="0">
      <count>0</count>
      <item_version>0</item_version>
    </aliasToEvent>
    <eventFilterEnabled>1</eventFilterEnabled>
    <eventNameFilter class_id="10" tracking_level="0" version="0">
      <count>1</count>
      <item_version>0</item_version>
      <item>HostGPSSample</item>
    </eventNameFilter>
    <eventTypeFilter class_id="11" tracking_level="0" version="0">
      <count>0</count>
      <item_version>0</item_version>
    </eventTypeFilter>
    <displayReceivedEvents>0</displayReceivedEvents>
    <fullyDescribeReceivedEvents>0</fullyDescribeReceivedEvents>
    <doNotDisplayList>
      <count>0</count>
      <item_version>0</item_version>
    </doNotDisplayList>
  </eventTransceiverSettings>
  <pmClientSettings class_id="12" tracking_level="0" version="0">
    <serviceName>PayloadManager</serviceName>
    <serviceKind/>
    <deviceAddedEventName>PayloadDeviceAdded</deviceAddedEventName>
    <deviceChangedEventName>PayloadDeviceChanged</deviceChangedEventName>
    <deviceRemovedEventName>PayloadDeviceRemoved</deviceRemovedEventName>
    <commonEventType>PayloadRegistrationEvent</commonEventType>
    <commonEventDomain>LiquidR-SMC</commonEventDomain>
    <payloadDescription class_id="13" tracking_level="0" version="2">
      <boardID>255</boardID>
      <taskID>241</taskID>
      <description>GPS event source and time updater</description>
      <ior>(This will be overwritten)</ior>
      <readableName>GPSListener</readableName>
      <descriptiveJSON/>
    </payloadDescription>
  </pmClientSettings>
</rootlevel>

```



## Appendix 2 - Sample Telemetry Configuration

The following listing is a sample configuration for the telemetry reported by PayloadInterface (PayloadInterface-telemetry-config.xml):

```
<?xml version="1.0"?>
<root>
  <publishers>
    <Element>
      <first boardID="0xFF" taskID="0xF0"/>
      <second>
        <key boardID="0xFF" taskID="0xF0"/>
        <payloadType value="0x0000007B"/>
        <eventNameToFormatCode>
          <Element>
            <first value="MYTELEMEVENT"/>
            <second value="0"/>
          </Element>
        </eventNameToFormatCode>
        <reportParams>
          <Element>
            <first value="0"/>
            <second maxSamples="2" targetAddress="2">
              <reportHeaderFormat>
                <fields>
                  <Element libField="Header91"/>
                  <Element libField="Raw" eventField="Telem-LengthToFollow"/>
                  <Element libField="Raw" eventField="Telem-PayloadType"/>
                  <Element libField="Raw" eventField="Telem-BoardID"/>
                  <Element libField="Raw" eventField="Telem-TaskID"/>
                  <Element libField="FC0BinaryNormal"/>
                  <Element libField="Raw" eventField="Telem-NumSamples"/>
                </fields>
              </reportHeaderFormat>
              <sampleHeaderFormat>
                <fields>
                  <Element libField="Raw" eventField="Telem-Latitude"/>
                  <Element libField="Raw" eventField="Telem-Longitude"/>
                  <Element libField="Raw" eventField="Telem-TimeSec"/>
                  <Element libField="DummySample"/>
                </fields>
              </sampleHeaderFormat>
            </second>
          </Element>
        </reportParams>
      </second>
    </Element>
    <Element>
      <first boardID="0xFF" taskID="0xF3"/>
      <second>
        <key boardID="0xFF" taskID="0xF3"/>
        <payloadType value="0x0000007D"/>
        <eventNameToFormatCode>
          <Element>
            <first value="BenthosModemRangeTelemetry"/>
            <second value="0"/>
          </Element>
        </eventNameToFormatCode>
        <reportParams>
          <Element>
            <first value="0"/>
            <second maxSamples="1" targetAddress="2">
              <reportHeaderFormat>
                <fields>
                  <Element libField="Header91"/>
                  <Element libField="Raw" eventField="Telem-LengthToFollow"/>
                  <Element libField="Raw" eventField="Telem-PayloadType"/>
                  <Element libField="Raw" eventField="Telem-BoardID"/>
                  <Element libField="Raw" eventField="Telem-TaskID"/>
                </fields>
              </reportHeaderFormat>
            </second>
          </Element>
        </reportParams>
      </second>
    </Element>
  </publishers>
</root>
```

```
<Element libField="FC0BinaryNormal"/>
<Element libField="RawINT8" eventField="Telem-NumSamples"/>
</fields>
</reportHeaderFormat>
<sampleHeaderFormat>
  <fields>
    <Element libField="Raw" eventField="Telem-Latitude"/>
    <Element libField="Raw" eventField="Telem-Longitude"/>
    <Element libField="Raw" eventField="Telem-TimeSec"/>
    <Element libField="Raw" eventField="ATRSample"/>
    <Element libField="Raw" eventField="ATXSample0"/>
    <Element libField="Raw" eventField="ATXSample1"/>
    <Element libField="Raw" eventField="ATXSample2"/>
    <Element libField="Raw" eventField="ATXSample3"/>
    <Element libField="Raw" eventField="ATXSample4"/>
    <Element libField="Raw" eventField="ATXSample5"/>
  </fields>
</sampleHeaderFormat>
</second>
</Element>
</reportParams>
</second>
</Element>
<Element>
  <first boardID="0xFF" taskID="0x12"/>
  <second>
    <key boardID="0xFF" taskID="0x12"/>
    <payloadType value="0x0000007B"/>
    <eventNameToFormatCode>
      <Element>
        <first value="EcoPuckSample"/>
        <second value="0"/>
      </Element>
    </eventNameToFormatCode>
    <reportParams>
      <Element>
        <first value="0"/>
        <second maxSamples="2" targetAddress="2">
          <reportHeaderFormat>
            <fields>
              <Element libField="Header91"/>
              <Element libField="Raw" eventField="Telem-LengthToFollow"/>
              <Element libField="Raw" eventField="Telem-PayloadType"/>
              <Element libField="Raw" eventField="Telem-BoardID"/>
              <Element libField="Raw" eventField="Telem-TaskID"/>
              <Element libField="FC0BinaryNormal"/>
              <Element libField="RawINT8" eventField="Telem-NumSamples"/>
            </fields>
          </reportHeaderFormat>
          <sampleHeaderFormat>
            <fields>
              <Element libField="Raw" eventField="Telem-Latitude"/>
              <Element libField="Raw" eventField="Telem-Longitude"/>
              <Element libField="Raw" eventField="Telem-TimeSec"/>
              <Element libField="Raw" eventField="col1"/>
              <Element libField="Raw" eventField="col2"/>
              <Element libField="Raw" eventField="col3"/>
              <Element libField="Raw" eventField="col4"/>
              <Element libField="Raw" eventField="col5"/>
              <Element libField="Raw" eventField="col6"/>
              <Element libField="Raw" eventField="col7"/>
              <Element libField="Raw" eventField="col8"/>
              <Element libField="Raw" eventField="col9"/>
            </fields>
          </sampleHeaderFormat>
        </second>
      </Element>
    </reportParams>
  </second>
</Element>
</Element>
```

```
<first boardID="0xFF" taskID="0x13"/>
<second>
  <key boardID="0xFF" taskID="0x13"/>
  <payloadType value="0x0000007C"/>
  <eventNameToFormatCode>
    <Element>
      <first value="CStarSample"/>
      <second value="0"/>
    </Element>
  </eventNameToFormatCode>
  <reportParams>
    <Element>
      <first value="0"/>
      <second maxSamples="2" targetAddress="2">
        <reportHeaderFormat>
          <fields>
            <Element libField="Header91"/>
            <Element libField="Raw" eventField="Telem-LengthToFollow"/>
            <Element libField="Raw" eventField="Telem-PayloadType"/>
            <Element libField="Raw" eventField="Telem-BoardID"/>
            <Element libField="Raw" eventField="Telem-TaskID"/>
            <Element libField="FC0BinaryNormal"/>
            <Element libField="RawINT8" eventField="Telem-NumSamples"/>
          </fields>
        </reportHeaderFormat>
        <sampleHeaderFormat>
          <fields>
            <Element libField="Raw" eventField="Telem-Latitude"/>
            <Element libField="Raw" eventField="Telem-Longitude"/>
            <Element libField="Raw" eventField="Telem-TimeSec"/>
            <Element libField="Raw" eventField="col1"/>
            <Element libField="Raw" eventField="col2"/>
            <Element libField="Raw" eventField="col3"/>
            <Element libField="Raw" eventField="col4"/>
            <Element libField="Raw" eventField="col5"/>
            <Element libField="Raw" eventField="col6"/>
          </fields>
        </sampleHeaderFormat>
      </second>
    </Element>
  </reportParams>
</second>
</Element>
<Element>
  <first boardID="0xFF" taskID="0x22"/>
  <second>
    <key boardID="0xFF" taskID="0x22"/>
    <payloadType value="0x0000007E"/>
    <eventNameToFormatCode>
      <Element>
        <first value="MOSESampleComplete"/>
        <second value="0"/>
      </Element>
    </eventNameToFormatCode>
    <reportParams>
      <Element>
        <first value="0"/>
        <second maxSamples="1" targetAddress="2">
          <reportHeaderFormat>
            <fields>
              <Element libField="Header91"/>
              <Element libField="Raw" eventField="Telem-LengthToFollow"/>
              <Element libField="Raw" eventField="Telem-PayloadType"/>
              <Element libField="Raw" eventField="Telem-BoardID"/>
              <Element libField="Raw" eventField="Telem-TaskID"/>
              <Element libField="FC0BinaryNormal"/>
              <Element libField="RawINT8" eventField="Telem-NumSamples"/>
            </fields>
          </reportHeaderFormat>
          <sampleHeaderFormat>
            <fields>
```

```
<Element libField="Raw" eventField="Telem-Latitude"/>
<Element libField="Raw" eventField="Telem-Longitude"/>
<Element libField="Raw" eventField="Telem-TimeSec"/>
<Element libField="Raw" eventField="MOSECompleteEvent_Hs"/>
<Element libField="Raw" eventField="MOSECompleteEvent_Tav"/>
<Element libField="Raw" eventField="MOSECompleteEvent_Tp"/>
<Element libField="Raw" eventField="MOSECompleteEvent_Dirp"/>
eventField="MOSECompleteEvent_NumSpectra"/>
<Element libField="Raw"
eventField="MOSECompleteEvent_SamplesPerSpectrum"/>
<Element libField="Raw"
eventField="MOSECompleteEvent_SampleGaps"/>
</fields>
</sampleHeaderFormat>
</second>
</Element>
</reportParams>
</second>
</Element>
<Element>
<first boardID="0xFF" taskID="0x15"/>
<second>
<key boardID="0xFF" taskID="0x15"/>
<payloadType value="0x0000007F"/>
<eventNameToFormatCode>
<Element>
<first value="VR2CTagReport"/>
<second value="0"/>
</Element>
</eventNameToFormatCode>
<reportParams>
<Element>
<first value="0"/>
<second maxSamples="1" targetAddress="2">
<reportHeaderFormat>
<fields>
<Element libField="Header91"/>
<Element libField="Raw" eventField="Telem-LengthToFollow"/>
<Element libField="Raw" eventField="Telem-PayloadType"/>
<Element libField="Raw" eventField="Telem-BoardID"/>
<Element libField="Raw" eventField="Telem-TaskID"/>
<Element libField="FC0ASCIINormal"/>
<Element libField="RawINT8" eventField="Telem-NumSamples"/>
</fields>
</reportHeaderFormat>
<sampleHeaderFormat>
<fields>
<Element libField="Raw" eventField="Telem-Latitude"/>
<Element libField="Raw" eventField="Telem-Longitude"/>
<Element libField="Raw" eventField="Telem-TimeSec"/>
<Element libField="Raw" eventField="StringEvent-String"/>
</fields>
</sampleHeaderFormat>
</second>
</Element>
</reportParams>
</second>
</Element>
<Element>
<first boardID="0xFF" taskID="0x17"/>
<second>
<key boardID="0xFF" taskID="0x17"/>
<payloadType value="0x00000060"/>
<eventNameToFormatCode>
<Element>
<first value="C3SampleHeader"/>
<second value="0"/>
</Element>
<Element>
<first value="C3SampleData"/>
```

```

        <second value="1"/>
    </Element>
</eventNameToFormatCode>
<reportParams>
    <Element>
        <first value="0"/>
        <second maxSamples="1" targetAddress="2">
            <reportHeaderFormat>
                <fields>
                    <Element libField="Header91"/>
                    <Element libField="Raw" eventField="Telem-LengthToFollow"/>
                    <Element libField="Raw" eventField="Telem-PayloadType"/>
                    <Element libField="Raw" eventField="Telem-BoardID"/>
                    <Element libField="Raw" eventField="Telem-TaskID"/>
                    <Element libField="FC0ASCIINormal"/>
                    <Element libField="RawINT8" eventField="Telem-NumSamples"/>
                </fields>
            </reportHeaderFormat>
            <sampleHeaderFormat>
                <fields>
                    <Element libField="Raw" eventField="Telem-Latitude"/>
                    <Element libField="Raw" eventField="Telem-Longitude"/>
                    <Element libField="Raw" eventField="Telem-TimeSec"/>
                    <Element libField="Raw" eventField="C3Event_Header"/>
                </fields>
            </sampleHeaderFormat>
        </second>
    </Element>
    <Element>
        <first value="1"/>
        <second maxSamples="7" targetAddress="2">
            <reportHeaderFormat>
                <fields>
                    <Element libField="Header91"/>
                    <Element libField="Raw" eventField="Telem-LengthToFollow"/>
                    <Element libField="Raw" eventField="Telem-PayloadType"/>
                    <Element libField="Raw" eventField="Telem-BoardID"/>
                    <Element libField="Raw" eventField="Telem-TaskID"/>
                    <Element libField="FC9BinaryNormal"/>
                    <Element libField="RawINT8" eventField="Telem-NumSamples"/>
                </fields>
            </reportHeaderFormat>
            <sampleHeaderFormat>
                <fields>
                    <Element libField="Raw" eventField="Telem-Latitude"/>
                    <Element libField="Raw" eventField="Telem-Longitude"/>
                    <Element libField="Raw" eventField="Telem-TimeSec"/>
                    <Element libField="Raw" eventField="C3Event_CompressedData"/>
                </fields>
            </sampleHeaderFormat>
        </second>
    </Element>
</reportParams>
</second>
</Element>
<Element>
    <first boardID="0xFF" taskID="0x10"/>
    <second>
        <key boardID="0xFF" taskID="0x10"/>
        <payloadType value="0x00000080"/>
        <eventNameToFormatCode>
            <Element>
                <first value="ADCPSampleHeaders"/>
                <second value="0"/>
            </Element>
            <Element>
                <first value="ADCPSampleB"/>
                <second value="1"/>
            </Element>
            <Element>
                <first value="ADCPSampleC2"/>

```

```
<second value="2"/>
</Element>
<Element>
  <first value="ADCPSampleC4"/>
  <second value="3"/>
</Element>
</eventNameToFormatCode>
<reportParams>
  <Element>
    <first value="0"/>
    <second maxSamples="1" targetAddress="2">
      <reportHeaderFormat>
        <fields>
          <Element libField="Header91"/>
          <Element libField="Raw" eventField="Telem-LengthToFollow"/>
          <Element libField="Raw" eventField="Telem-PayloadType"/>
          <Element libField="Raw" eventField="Telem-BoardID"/>
          <Element libField="Raw" eventField="Telem-TaskID"/>
          <Element libField="FC2BinaryNormal"/>
          <Element libField="RawINT8" eventField="Telem-NumSamples"/>
        </fields>
      </reportHeaderFormat>
      <sampleHeaderFormat>
        <fields>
          <Element libField="Raw" eventField="Telem-Latitude"/>
          <Element libField="Raw" eventField="Telem-Longitude"/>
          <Element libField="Raw" eventField="Telem-TimeSec"/>
          <Element libField="Raw" eventField="ADCPEvent-Binary"/>
        </fields>
      </sampleHeaderFormat>
    </second>
  </Element>
  <Element>
    <first value="1"/>
    <second maxSamples="1" targetAddress="2">
      <reportHeaderFormat>
        <fields>
          <Element libField="Header91"/>
          <Element libField="Raw" eventField="Telem-LengthToFollow"/>
          <Element libField="Raw" eventField="Telem-PayloadType"/>
          <Element libField="Raw" eventField="Telem-BoardID"/>
          <Element libField="Raw" eventField="Telem-TaskID"/>
          <Element libField="FC3BinaryNormal"/>
          <Element libField="RawINT8" eventField="Telem-NumSamples"/>
        </fields>
      </reportHeaderFormat>
      <sampleHeaderFormat>
        <fields>
          <Element libField="Raw" eventField="Telem-Latitude"/>
          <Element libField="Raw" eventField="Telem-Longitude"/>
          <Element libField="Raw" eventField="Telem-TimeSec"/>
          <Element libField="Raw" eventField="ADCPEvent-Binary"/>
        </fields>
      </sampleHeaderFormat>
    </second>
  </Element>
  <Element>
    <first value="2"/>
    <second maxSamples="1" targetAddress="2">
      <reportHeaderFormat>
        <fields>
          <Element libField="Header91"/>
          <Element libField="Raw" eventField="Telem-LengthToFollow"/>
          <Element libField="Raw" eventField="Telem-PayloadType"/>
          <Element libField="Raw" eventField="Telem-BoardID"/>
          <Element libField="Raw" eventField="Telem-TaskID"/>
          <Element libField="FC4BinaryNormal"/>
          <Element libField="RawINT8" eventField="Telem-NumSamples"/>
        </fields>
      </reportHeaderFormat>
      <sampleHeaderFormat>
```

```

        <fields>
        <Element libField="Raw" eventField="Telem-Latitude"/>
        <Element libField="Raw" eventField="Telem-Longitude"/>
        <Element libField="Raw" eventField="Telem-TimeSec"/>
        <Element libField="Raw" eventField="ADCPEvent-Binary"/>
        </fields>
    </sampleHeaderFormat>
</second>
</Element>
<Element>
    <first value="3"/>
    <second maxSamples="1" targetAddress="2">
        <reportHeaderFormat>
            <fields>
                <Element libField="Header91"/>
                <Element libField="Raw" eventField="Telem-LengthToFollow"/>
                <Element libField="Raw" eventField="Telem-PayloadType"/>
                <Element libField="Raw" eventField="Telem-BoardID"/>
                <Element libField="Raw" eventField="Telem-TaskID"/>
                <Element libField="FC5BinaryNormal"/>
                <Element libField="RawINT8" eventField="Telem-NumSamples"/>
            </fields>
        </reportHeaderFormat>
        <sampleHeaderFormat>
            <fields>
                <Element libField="Raw" eventField="Telem-Latitude"/>
                <Element libField="Raw" eventField="Telem-Longitude"/>
                <Element libField="Raw" eventField="Telem-TimeSec"/>
                <Element libField="Raw" eventField="ADCPEvent-Binary"/>
            </fields>
        </sampleHeaderFormat>
    </second>
</Element>
</reportParams>
</second>
</Element>
<Element>
    <first boardID="0xFF" taskID="0x40"/>
    <second>
        <key boardID="0xFF" taskID="0x40"/>
        <payloadType value="0x000000FE"/>
        <eventNameToFormatCode>
            <Element>
                <first value="LWTelem1"/>
                <second value="0"/>
            </Element>
        </eventNameToFormatCode>
        <reportParams>
            <Element>
                <first value="0"/>
                <second maxSamples="1" targetAddress="2">
                    <reportHeaderFormat>
                        <fields>
                            <Element libField="Header91"/>
                            <Element libField="Raw" eventField="Telem-LengthToFollow"/>
                            <Element libField="Raw" eventField="Telem-PayloadType"/>
                            <Element libField="Raw" eventField="Telem-BoardID"/>
                            <Element libField="Raw" eventField="Telem-TaskID"/>
                            <Element libField="FC0ASCIINormal"/>
                            <Element libField="RawINT8" eventField="Telem-NumSamples"/>
                        </fields>
                    </reportHeaderFormat>
                    <sampleHeaderFormat>
                        <fields>
                            <Element libField="Raw" eventField="Telem-Latitude"/>
                            <Element libField="Raw" eventField="Telem-Longitude"/>
                            <Element libField="Raw" eventField="Telem-TimeSec"/>
                            <Element libField="Raw" eventField="StringMessage"/>
                        </fields>
                    </sampleHeaderFormat>
                </second>
            </Element>
        </reportParams>
    </second>
</Element>

```

```
</Element>
</reportParams>
</second>
</Element>
<Element>
  <first boardID="0xFF" taskID="0x18"/>
  <second>
    <key boardID="0xFF" taskID="0x18"/>
    <payloadType value="0x00000083"/>
    <eventNameToFormatCode>
      <Element>
        <first value="WHCycleReport"/>
        <second value="0"/>
      </Element>
    </eventNameToFormatCode>
    <reportParams>
      <Element>
        <first value="0"/>
        <second maxSamples="1" targetAddress="2">
          <reportHeaderFormat>
            <fields>
              <Element libField="Header91"/>
              <Element libField="Raw" eventField="Telem-LengthToFollow"/>
              <Element libField="Raw" eventField="Telem-PayloadType"/>
              <Element libField="Raw" eventField="Telem-BoardID"/>
              <Element libField="Raw" eventField="Telem-TaskID"/>
              <Element libField="FC0ASCIINormal"/>
              <Element libField="RawINT8" eventField="Telem-NumSamples"/>
            </fields>
          </reportHeaderFormat>
          <sampleHeaderFormat>
            <fields>
              <Element libField="Raw" eventField="Telem-Latitude"/>
              <Element libField="Raw" eventField="Telem-Longitude"/>
              <Element libField="Raw" eventField="Telem-TimeSec"/>
              <Element libField="Raw" eventField="StringEvent-String"/>
            </fields>
          </sampleHeaderFormat>
        </second>
      </Element>
    </reportParams>
  </second>
</Element>
<Element>
  <first boardID="0xFF" taskID="0x19"/>
  <second>
    <key boardID="0xFF" taskID="0x19"/>
    <payloadType value="0x00000087"/>
    <eventNameToFormatCode>
      <Element>
        <first value="LISSTSummaryDataEvent"/>
        <second value="0"/>
      </Element>
    </eventNameToFormatCode>
    <reportParams>
      <Element>
        <first value="0"/>
        <second maxSamples="5" targetAddress="2">
          <reportHeaderFormat>
            <fields>
              <Element libField="Header91"/>
              <Element libField="Raw" eventField="Telem-LengthToFollow"/>
              <Element libField="Raw" eventField="Telem-PayloadType"/>
              <Element libField="Raw" eventField="Telem-BoardID"/>
              <Element libField="Raw" eventField="Telem-TaskID"/>
              <Element libField="FC0BinaryNormal"/>
              <Element libField="RawINT8" eventField="Telem-NumSamples"/>
            </fields>
          </reportHeaderFormat>
          <sampleHeaderFormat>
            <fields>
```

```

        <Element libField="Raw" eventField="Telem-Latitude"/>
        <Element libField="Raw" eventField="Telem-Longitude"/>
        <Element libField="Raw" eventField="Telem-TimeSec"/>
        <Element libField="Raw"
eventField="LISSTSummaryDataEvent_Volume"/>
        <Element libField="Raw"
eventField="LISSTSummaryDataEvent_Size"/>
        <Element libField="Raw"
eventField="LISSTSummaryDataEvent_SampleCount"/>
        <Element libField="Raw"
eventField="LISSTSummaryDataEvent_SerialNumber"/>
        <Element libField="Raw"
eventField="LISSTSummaryDataEvent_BackgroundFileDateLength"/>
        <Element libField="Raw"
eventField="LISSTSummaryDataEvent_BackgroundFileDate"/>
        </fields>
    </sampleHeaderFormat>
</second>
</Element>
</reportParams>
</second>
</Element>
<Element>
    <first boardID="0xFF" taskID="0x1D"/>
    <second>
        <key boardID="0xFF" taskID="0x1D"/>
        <payloadType value="0x0000008C"/>
        <eventNameToFormatCode>
            <Element>
                <first value="MagnetometerRawData"/>
                <second value="0"/>
            </Element>
        </eventNameToFormatCode>
        <reportParams>
            <Element>
                <first value="0"/>
                <second maxSamples="2" targetAddress="2">
                    <reportHeaderFormat>
                        <fields>
                            <Element libField="Header91"/>
                            <Element libField="Raw" eventField="Telem-LengthToFollow"/>
                            <Element libField="Raw" eventField="Telem-PayloadType"/>
                            <Element libField="Raw" eventField="Telem-BoardID"/>
                            <Element libField="Raw" eventField="Telem-TaskID"/>
                            <Element libField="FC1BinaryNormal"/>
                            <Element libField="RawINT8" eventField="Telem-NumSamples"/>
                        </fields>
                    </reportHeaderFormat>
                    <sampleHeaderFormat>
                        <fields>
                            <Element libField="Raw" eventField="Telem-Latitude"/>
                            <Element libField="Raw" eventField="Telem-Longitude"/>
                            <Element libField="Raw" eventField="Telem-TimeSec"/>
                            <Element libField="Raw" eventField="Filtered Mag Field (nT)"/>
                            <Element libField="Raw" eventField="Date YY.JJJ"/>
                            <Element libField="Raw" eventField="Time HH:MM:SS.s"/>
                            <Element libField="Raw" eventField="Mag Field (nT)"/>
                            <Element libField="Raw" eventField="Signal Strength"/>
                            <Element libField="Raw" eventField="Depth (m)"/>
                            <Element libField="Raw" eventField="Leak"/>
                            <Element libField="Raw" eventField="Measurement Time (ms)"/>
                            <Element libField="Raw" eventField="Signal Quality"/>
                        </fields>
                    </sampleHeaderFormat>
                </second>
            </Element>
        </reportParams>
    </second>
</Element>
<Element>
    <first boardID="0xFF" taskID="0x1E"/>

```

```

<second>
  <key boardID="0xFF" taskID="0x1E"/>
  <payloadType value="0x00000091"/>
  <eventNameToFormatCode>
    <Element>
      <first value="VemcoVM4Message"/>
      <second value="0"/>
    </Element>
    <Element>
      <first value="VemcoVM4Realtime"/>
      <second value="1"/>
    </Element>
    <Element>
      <first value="VemcoVM4LocalHealth"/>
      <second value="2"/>
    </Element>
    <Element>
      <first value="VemcoVM4RemoteHealth"/>
      <second value="3"/>
    </Element>
    <Element>
      <first value="VemcoVM4DailyLocalHealth"/>
      <second value="4"/>
    </Element>
    <Element>
      <first value="VemcoVM4DailyRemoteHealth"/>
      <second value="5"/>
    </Element>
  </eventNameToFormatCode>
  <reportParams>
    <Element>
      <first value="0"/>
      <second maxSamples="1" targetAddress="2">
        <reportHeaderFormat>
          <fields>
            <Element libField="Header91"/>
            <Element libField="Raw" eventField="Telem-LengthToFollow"/>
            <Element libField="Raw" eventField="Telem-PayloadType"/>
            <Element libField="Raw" eventField="Telem-BoardID"/>
            <Element libField="Raw" eventField="Telem-TaskID"/>
            <Element libField="FC0ASCIINormal"/>
            <Element libField="RawINT8" eventField="Telem-NumSamples"/>
          </fields>
        </reportHeaderFormat>
        <sampleHeaderFormat>
          <fields>
            <Element libField="Raw" eventField="Telem-Latitude"/>
            <Element libField="Raw" eventField="Telem-Longitude"/>
            <Element libField="Raw" eventField="Telem-TimeSec"/>
            <Element libField="Raw" eventField="VemcoVM4Event_Header"/>
          </fields>
        </sampleHeaderFormat>
      </second>
    </Element>
    <Element>
      <first value="1"/>
      <second maxSamples="8" maxRetainTime="300" targetAddress="2">
        <reportHeaderFormat>
          <fields>
            <Element libField="Header91"/>
            <Element libField="Raw" eventField="Telem-LengthToFollow"/>
            <Element libField="Raw" eventField="Telem-PayloadType"/>
            <Element libField="Raw" eventField="Telem-BoardID"/>
            <Element libField="Raw" eventField="Telem-TaskID"/>
            <Element libField="FC1BinaryNormal"/>
            <Element libField="RawINT8" eventField="Telem-NumSamples"/>
          </fields>
        </reportHeaderFormat>
        <sampleHeaderFormat>
          <fields>
            <Element libField="Raw" eventField="Telem-Latitude"/>

```

```
<Element libField="Raw" eventField="Telem-Longitude"/>
<Element libField="Raw" eventField="Telem-TimeSec"/>
<Element libField="Raw" eventField="VemcoVM4Event_Realtime"/>
</fields>
</sampleHeaderFormat>
</second>
</Element>
<Element>
  <first value="2"/>
  <second maxSamples="1" targetAddress="2">
    <reportHeaderFormat>
      <fields>
        <Element libField="Header91"/>
        <Element libField="Raw" eventField="Telem-LengthToFollow"/>
        <Element libField="Raw" eventField="Telem-PayloadType"/>
        <Element libField="Raw" eventField="Telem-BoardID"/>
        <Element libField="Raw" eventField="Telem-TaskID"/>
        <Element libField="FC2BinaryNormal"/>
        <Element libField="RawINT8" eventField="Telem-NumSamples"/>
      </fields>
    </reportHeaderFormat>
    <sampleHeaderFormat>
      <fields>
        <Element libField="Raw" eventField="Telem-Latitude"/>
        <Element libField="Raw" eventField="Telem-Longitude"/>
        <Element libField="Raw" eventField="Telem-TimeSec"/>
        <Element libField="Raw" eventField="VemcoVM4Event_Health"/>
      </fields>
    </sampleHeaderFormat>
  </second>
</Element>
<Element>
  <first value="3"/>
  <second maxSamples="1" targetAddress="2">
    <reportHeaderFormat>
      <fields>
        <Element libField="Header91"/>
        <Element libField="Raw" eventField="Telem-LengthToFollow"/>
        <Element libField="Raw" eventField="Telem-PayloadType"/>
        <Element libField="Raw" eventField="Telem-BoardID"/>
        <Element libField="Raw" eventField="Telem-TaskID"/>
        <Element libField="FC3BinaryNormal"/>
        <Element libField="RawINT8" eventField="Telem-NumSamples"/>
      </fields>
    </reportHeaderFormat>
    <sampleHeaderFormat>
      <fields>
        <Element libField="Raw" eventField="Telem-Latitude"/>
        <Element libField="Raw" eventField="Telem-Longitude"/>
        <Element libField="Raw" eventField="Telem-TimeSec"/>
        <Element libField="Raw" eventField="VemcoVM4Event_Health"/>
      </fields>
    </sampleHeaderFormat>
  </second>
</Element>
<Element>
  <first value="4"/>
  <second maxSamples="1" targetAddress="2">
    <reportHeaderFormat>
      <fields>
        <Element libField="Header91"/>
        <Element libField="Raw" eventField="Telem-LengthToFollow"/>
        <Element libField="Raw" eventField="Telem-PayloadType"/>
        <Element libField="Raw" eventField="Telem-BoardID"/>
        <Element libField="Raw" eventField="Telem-TaskID"/>
        <Element libField="FC4BinaryNormal"/>
        <Element libField="RawINT8" eventField="Telem-NumSamples"/>
      </fields>
    </reportHeaderFormat>
    <sampleHeaderFormat>
      <fields>
```

```

        <Element libField="Raw" eventField="Telem-Latitude"/>
        <Element libField="Raw" eventField="Telem-Longitude"/>
        <Element libField="Raw" eventField="Telem-TimeSec"/>
        <Element libField="Raw" eventField="VemcoVM4Event_Health"/>
    </fields>
</sampleHeaderFormat>
</second>
</Element>
<Element>
    <first value="5"/>
    <second maxSamples="1" targetAddress="2">
        <reportHeaderFormat>
            <fields>
                <Element libField="Header91"/>
                <Element libField="Raw" eventField="Telem-LengthToFollow"/>
                <Element libField="Raw" eventField="Telem-PayloadType"/>
                <Element libField="Raw" eventField="Telem-BoardID"/>
                <Element libField="Raw" eventField="Telem-TaskID"/>
                <Element libField="FC5BinaryNormal"/>
                <Element libField="RawINT8" eventField="Telem-NumSamples"/>
            </fields>
        </reportHeaderFormat>
        <sampleHeaderFormat>
            <fields>
                <Element libField="Raw" eventField="Telem-Latitude"/>
                <Element libField="Raw" eventField="Telem-Longitude"/>
                <Element libField="Raw" eventField="Telem-TimeSec"/>
                <Element libField="Raw" eventField="VemcoVM4Event_Health"/>
            </fields>
        </sampleHeaderFormat>
    </second>
</Element>
</reportParams>
</second>
</Element>
<Element>
    <first boardID="0xFF" taskID="0x25"/>
    <second>
        <key boardID="0xFF" taskID="0x25"/>
        <payloadType value="0x00000093"/>
        <eventNameToFormatCode>
            <Element>
                <first value="GPSWavesTelem"/>
                <second value="0"/>
            </Element>
        </eventNameToFormatCode>
        <reportParams>
            <Element>
                <first value="0"/>
                <second maxSamples="1" targetAddress="2">
                    <reportHeaderFormat>
                        <fields>
                            <Element libField="Header91"/>
                            <Element libField="Raw" eventField="Telem-LengthToFollow"/>
                            <Element libField="Raw" eventField="Telem-PayloadType"/>
                            <Element libField="Raw" eventField="Telem-BoardID"/>
                            <Element libField="Raw" eventField="Telem-TaskID"/>
                            <Element libField="FC0BinaryNormal"/>
                            <Element libField="RawINT8" eventField="Telem-NumSamples"/>
                        </fields>
                    </reportHeaderFormat>
                    <sampleHeaderFormat>
                        <fields>
                            <Element libField="Raw" eventField="Telem-Latitude"/>
                            <Element libField="Raw" eventField="Telem-Longitude"/>
                            <Element libField="Raw" eventField="Telem-TimeSec"/>
                            <Element libField="Raw" eventField="GPSWavesTelemEvent_Hs"/>
                            <Element libField="Raw" eventField="GPSWavesTelemEvent_Tp"/>
                            <Element libField="Raw" eventField="GPSWavesTelemEvent_Dp"/>
                            <Element libField="Raw"
eventField="GPSWavesTelemEvent_Samples"/>
                        </fields>
                    </sampleHeaderFormat>
                </second>
            </Element>
        </reportParams>
    </second>
</Element>

```

```
<Element libField="Raw" eventField="GPSWavesTelemEvent_Fs"/>
<Element libField="Raw"
eventField="GPSWavesTelemEvent_SampleGaps"/>
<Element libField="Raw" eventField="GPSWavesTelemEvent_Ta"/>
</fields>
</sampleHeaderFormat>
</second>
</Element>
</reportParams>
</second>
</Element>
<Element>
<first boardID="0xFF" taskID="0x26"/>
<second>
<key boardID="0xFF" taskID="0x26"/>
<payloadType value="0x00000070"/>
<eventNameToFormatCode>
<Element>
<first value="SBGPCTDSample"/>
<second value="8"/>
</Element>
<Element>
<first value="SBGPCTDSampleWithDO"/>
<second value="9"/>
</Element>
</eventNameToFormatCode>
<reportParams>
<Element>
<first value="8"/>
<second maxSamples="1" targetAddress="2">
<reportHeaderFormat>
<fields>
<Element libField="Header91"/>
<Element libField="Raw" eventField="Telem-LengthToFollow"/>
<Element libField="Raw" eventField="Telem-PayloadType"/>
<Element libField="Raw" eventField="Telem-BoardID"/>
<Element libField="Raw" eventField="Telem-TaskID"/>
<Element libField="FC8BinaryNormal"/>
<Element libField="RawINT8" eventField="Telem-NumSamples"/>
</fields>
</reportHeaderFormat>
<sampleHeaderFormat>
<fields>
<Element libField="Raw" eventField="Telem-Latitude"/>
<Element libField="Raw" eventField="Telem-Longitude"/>
<Element libField="Raw" eventField="Telem-TimeSec"/>
<Element libField="Raw" eventField="SBGPCTDSample_RawPressure"/>
<Element libField="Raw"
eventField="SBGPCTDSample_RawTemperature"/>
<Element libField="Raw"
eventField="SBGPCTDSample_RawConductivity"/>
</fields>
</sampleHeaderFormat>
</second>
</Element>
<Element>
<first value="9"/>
<second maxSamples="1" targetAddress="2">
<reportHeaderFormat>
<fields>
<Element libField="Header91"/>
<Element libField="Raw" eventField="Telem-LengthToFollow"/>
<Element libField="Raw" eventField="Telem-PayloadType"/>
<Element libField="Raw" eventField="Telem-BoardID"/>
<Element libField="Raw" eventField="Telem-TaskID"/>
<Element libField="FC9BinaryNormal"/>
<Element libField="RawINT8" eventField="Telem-NumSamples"/>
</fields>
</reportHeaderFormat>
<sampleHeaderFormat>
<fields>
```

```

        <Element libField="Raw" eventField="Telem-Latitude"/>
        <Element libField="Raw" eventField="Telem-Longitude"/>
        <Element libField="Raw" eventField="Telem-TimeSec"/>
        <Element libField="Raw" eventField="SBGPCTDSample_RawPressure"/>
        <Element libField="Raw"
eventField="SBGPCTDSample_RawTemperature"/>
        <Element libField="Raw"
eventField="SBGPCTDSample_RawConductivity"/>
        <Element libField="Raw"
eventField="SBGPCTDSample_RawDissolvedOxygen"/>
    </fields>
</sampleHeaderFormat>
</second>
</Element>
</reportParams>
</second>
</Element>
</publishers>
<fieldLibrary>
    <lib>
        <Element>
            <key value="Raw"/>
            <value content="CA_RAW"/>
        </Element>
        <Element>
            <key value="DummySample"/>
            <value content="CA_LITERAL">
                <literal type="STRING" value="[D]"/>
            </value>
        </Element>
        <Element>
            <key value="Header91"/>
            <value content="CA_LITERAL">
                <literal type="UINT8" value="0x91"/>
            </value>
        </Element>
        <Element>
            <key value="FC0ASCIINormal"/>
            <value content="CA_LITERAL">
                <literal type="UINT8" value="0x10"/>
            </value>
        </Element>
        <Element>
            <key value="FC0BinaryNormal"/>
            <value content="CA_LITERAL">
                <literal type="UINT8" value="0x00"/>
            </value>
        </Element>
        <Element>
            <key value="FC1BinaryNormal"/>
            <value content="CA_LITERAL">
                <literal type="UINT8" value="0x01"/>
            </value>
        </Element>
        <Element>
            <key value="FC2BinaryNormal"/>
            <value content="CA_LITERAL">
                <literal type="UINT8" value="0x02"/>
            </value>
        </Element>
        <Element>
            <key value="FC3BinaryNormal"/>
            <value content="CA_LITERAL">
                <literal type="UINT8" value="0x03"/>
            </value>
        </Element>
        <Element>
            <key value="FC4BinaryNormal"/>
            <value content="CA_LITERAL">
                <literal type="UINT8" value="0x04"/>
            </value>
        </Element>
    </lib>
</fieldLibrary>

```

```
</Element>
<Element>
  <key value="FC5BinaryNormal"/>
  <value content="CA_LITERAL">
    <literal type="UINT8" value="0x05"/>
  </value>
</Element>
<Element>
  <key value="FC6BinaryNormal"/>
  <value content="CA_LITERAL">
    <literal type="UINT8" value="0x06"/>
  </value>
</Element>
<Element>
  <key value="FC7BinaryNormal"/>
  <value content="CA_LITERAL">
    <literal type="UINT8" value="0x07"/>
  </value>
</Element>
<Element>
  <key value="FC8BinaryNormal"/>
  <value content="CA_LITERAL">
    <literal type="UINT8" value="0x08"/>
  </value>
</Element>
<Element>
  <key value="FC9BinaryNormal"/>
  <value content="CA_LITERAL">
    <literal type="UINT8" value="0x09"/>
  </value>
</Element>
<Element>
  <key value="RawINT8"/>
  <value content="CA_LENGTH1"/>
</Element>
</lib>
</fieldLibrary>
</root>
```

## History

---

| date       | changes made            | author         |
|------------|-------------------------|----------------|
| 02/01/2016 | Edits for 2.5.1 release | Andrew Trapani |