**myRailIO Client CLI command reference guide**

**CLI main principles:**

The myRailIOClient provides a CLI through Telnet or SSH (configurable). The CLI is based on the Arduino simpleCLI library and provides capabilities to view and modify objects through-out the myRailIO managed object tree - which we refer to as cli contexts. At any given point in time, the CLI points/operates on a particular context which provides the CLI capabilities for that specific CLI context/Managed object tree instance.  
There are three different categories of CLI commands:

* **Global CLI commands**: These CLI commands are available in all CLI contexts, but does not operate on the specific context, but on the global context. Example: “reboot”…
* **Common CLI commands**: These CLI commands are available in all CLI contexts and provides capabilities that are common for all CLI contexts, it operates on the CLI context (not the global context). Example: “get sysstate” or “get sysState”…
* **Context unique CLI commands**: These CLI commands are unique to each CLI context. Example: “get aspect” or “set aspect”…

There is a well-defined set of main CLI command methods across all CLI contexts which is globally defined, these cannot be changed or extended by CLI contexts. Examples: help, show, get, set….

The concept of CLI context path provides the capability to navigate across various CLI contexts/myRailIO management object model tree junctions. The CLI context can be changed by explicitly set the context by providing a absolute context definition: “set context {/absolutrContextPath}”, or by relative context definition: “set context {rellativeContextPath}”.

CLI commands can operate on other CLI contexts than what is the current CLI context by providing context routing information to the CLI command - “mainCmd [contextPath]firstArg [secondArg] [-nonPosflags][-nonPosArgs=][][]”… Example: “get ../lglink-1/sysname” or “get /./../lglink-1/sysname” or “get decoder-0/lglink-1/sysname”.  
CLI contextPath in large follows the syntax and semantics of Posix/Unix file definitions.

**High-level myRailIO class/object model**The CLI context model in large follows the myRailIO class/object model which is shown below:

Picture

**Available Main commands:**

**myRailIO limits the main command to the following:**

**Cmd: CmdCategory:**

help Global CLI command

reboot Global CLI command

show Global-, common-, and context unique

get Global-, common-, and context unique

set Global-, common-, and context unique

unset Global-, common-, and context unique

add Not implemented

delete Not implemented

copy Not implemented

paste Not implemented

move Not implemented

start Global-, common-, and context unique

stop Global-, common-, and context unique

restart Not implemented

**CLIContextPath**

The CLIContextPath is a prefix to the second positional argument CLI cmd method, and in large follows the Posix/Unix file navigation methods. However, the pure CLI hierarchical context index-routing is complemented with an unstructured CLI context unique SystemName routing.

contextPath:[..|{{contextName}{-}{contextIndex}}|{contextSystemName}[{/}{contextPath}]{/}]

**CLI context names**

**Following CLI context names apply:**

* (decoder-{instanceIndex}(|({decoderSystemName})
* (lightgrouplink-{instanceIndex}) | (lgrouplink-{instanceIndex}) |  
  (lglink-{instanceIndex}) | {lgLinkSystemName}
* (lightgroup-{instanceIndex}) | (lg|lgMast-{instanceIndex}) | {lgSystemName}
* (satelitelink-{instanceIndex}) | (satlink-{instanceIndex}) | {satlinkSystemName}
* (satelite-{instanceIndex}) | (sat-{instanceIndex}) | {satSystemName}
* (actuator-{instanceIndex}) | (act-{instanceIndex}) | {actSystemName}
* (sensor-{instanceIndex}) | (sens-{instanceIndex}) | {sensSystemName}

**CLI context routing - example**

# get context

decoder-0/sateliteLink-1/satelite-0

# get ../satelite-1/sysstate

decoder-0/sateliteLink-1/satelite-1 sysstate: WORKING

**Global myRailIO client commands (Available in all contexts, operating on the global context)**

help [command [subcommand]] <<<HELP NEEDS WORK>>>

reboot <<<VERIFIED IN ROOT CONTEXT>>>

get context <<<VERIFIED IN ROOT CONTEXT >>>

set context {context} <<<VERIFIED IN ROOT CONTEXT, BUT COMMANDS NOT ACCEPTED AFTERWARDS>>>

get uptime <<<VERIFIED IN ROOT CONTEXT >>>

get network [-ssid] [-bssid] [-channel] [-auth] [-rssi] [-mac] [-hostname] [-address] [-mask] [-gw]   
[-dns] [-opstate] <<<VERIFIED IN ROOT CONTEXT >>>

**Time:**Doesn’t work/not implemented

* ntpport
* ntpdhcp
* daylight saving
* setting of syncmode

set network [-addr {address}] [-mask {mask}] [-gw {gateway}] [-dns {dns}]

add time -ntpserver {server URI} [-ntpport{port}]<<<VERIFIED IN ROOT CONTEXT, PORT IGNORED>>>

delete time -ntpserver{server URI}

start time -ntpclient | -ntpdhcp <<<VERIFIED IN ROOT CONTEXT, NTPDHCP NOT IMPLEMENTED >>>

stop time -ntpclient | -ntpdhcp <<<VERIFIED IN ROOT CONTEXT, NTPDHCP NOT IMPLEMENTED >>>

set time [-timeofday/tod{YYYY-MM-DDTHH:MM:SS] [-epochtime {epochtime}] [-timezone {std unix tz}] [-daylightsaving {“true”|”false”}]

get time [-timeofday/tod [-utc]] [-utc] [-timezone] [-daylightsaving] [-ntpdhcp] [-ntpservers] [-ntpsycstatus] [-ntpsyncmode] <<<VERIFIED IN ROOT CONTEXT, DAYLIGHT SAVING DOES NOT WORK, NTPDHCP NOT IMPLEMENTED >>>

show time

**MQTT:**

get mqtt [-uri] [-port] [-clientid] [-qos] [-keepalive] [-ping] [-maxlatency] [-meanlatency] [-overruns] [-opstate] [-subscriptions] <<<VERIFIED IN ROOT CONTEXT >>> (Stats not OK)

show mqtt (same as get mqtt) <<<VERIFIED IN ROOT CONTEXT >>> (Stats not OK)

clear mqtt -maxlatency|-overruns ) <<<VERIFIED IN ROOT CONTEXT >>> (Stats not OK)

set mqtt [-broker {broker}] | [-port {port}] [-clientId {clientId}] [-qos {0|1|2}]

**LOG:**

get log (ACTUALLY COMMON ONCE WE INTRODUCE CONTEXT AWARE VERBOSITY) <<<VERIFIED IN ROOT CONTEXT >>

set log/verbosity verbosity {fatal|error|notice|verbose}

get rsys/log

set rsys/log {uri}

show log [-size {logLines}] [-append]

**MEMORY:**

get memory [-internal] [-total] [-available] [-used] [-watermark] [-average {period\_S}] [-trend {period\_s}] [-maxblock] <<<VERIFIED IN ROOT CONTEXT>>>

show memory (same as get memory) <<<VERIFIED IN ROOT CONTEXT – SHOULD ALSO SHOW INTERNAL>>>

start memory [-internal|-external|-default] -allocate{bytes} <<<VERIFIED IN ROOT CONTEXT>>>

stop memory -allocate <<<VERIFIED IN ROOT CONTEXT>>>

start pm [-cpu] [-heap] [-stack]

stop pm [-cpu] [-heap] [-stack]

show proc/ess/stats [-proc/ess/name {process name}]

show cpu/stats [-core {0|1}] [-period {1|10|60}]

show heap/stats [-core {0|1}] [period {1|10|60}] [-max]

show stack/stats proc/ess/name [-core {0|1}] [period {1|10|60}] [-max]

show mo/topology,mo/tree [-base/mo { contextPath}]

**Common commands (Available in all contexts operating on the given context)**

get [contextPath]sysstate

set [contextPath]sysstate{DISCONNECTED|UNDISCOVERED|UNCONFIGURED|DISABLED|  
 UNAVAILABLE|INTFAIL|CBL|UNUSED}

unset [contextPath]sysstate statebit {DISCONNECTED|UNDISCOVERED|UNCONFIGURED|DISABLED|  
 UNAVAILABLE|INTFAIL|CBL|UNUSED}

get context [-childs [prefix {prefix filter}] [-parent]

set context {contextpath}

get [contextPath] sysname

get [contextPath] usrname,username

set [contextPath] usrname,username {username}

get [contextPath]desc/ription

set [contextPath]desc/ription {description}

**Decoder CLI commands (Available in the decoder context)**

-

**LightgroupLink CLI commands (Available in the LightgroupLink context)**

get[contextPath]link/no

set [contextPath]link/no {linkno}

get [contextPath]stats [-over/runs] [-meanlat/ency] [-maxlat/ency] [-meanrun/time] [-maxrun/time]

clear [contextPath]stats

**Lightgroup CLI commands (Available in the Lightgroup context)**

**Lightgroup common commands**

get [contextPath]addr/ess

set [contextPath]addr/ess address

get [contextPath]property [-propid,property/id {propertyId}]

set [contextPath]property -propid,property/id {propertyId}

**Signal mast Lightgroup extentions (Available if Lightgroup context is a Signal mast)**

get [contextPath]noofleds

get [contextPath]aspect

set [contextPath]aspect aspect

**SateliteLink CLI commands (Available in the SateliteLink context)**

get[contextPath]link/no

set [contextPath]link/no {linkno}

get [contextPath]stats [-Txover/runs] [-Txunder/runs] [-Rxover/runs] [-Rxunder/runs] [-RxCrcErrs]  
[-RxSymbolErrs] [-maxlat/ency] [-meanrun/time] [-maxrun/time] [-scantimingviolations]

clear [contextPath]stats

**Satelite CLI commands (Available in the Satelite context)**

**Actuator CLI commands (Available in the Actuator context)**

**Sensor CLI commands (Available in the Sensor context)**