

Scripting Reference

MotionNode Version 1.4

<http://www.motionnode.com/>

Copyright © 2011 GLI Interactive LLC. All rights reserved.

The coded instructions, statements, computer programs, and/or related material (collectively the “Data”) in these files contain unpublished information proprietary to GLI Interactive LLC, which is protected by US federal copyright law and by international treaties.

The Data may not be disclosed or distributed to third parties, in whole or in part, without the prior written consent of GLI Interactive LLC.

The Data is provided “as is” without express or implied warranty, and with no claim as to its suitability for any purpose.

Contents

1	Introduction	4
2	Getting Started	4
2.1	What is Lua?	4
2.2	Interactive Console	4
2.3	Script File	5
2.4	Basic Commands	6
3	Module Reference	8
3.1	node	8
	close	9
	configuration	10
	connect	10
	connected	10
	define_identity_pose	10
	delay_to_sample_rate	10
	erase	11
	export	11
	export_stream	11
	get_node	11
	get_node_by_id	12
	get_node_by_key	12
	get_preview	12
	get_raw	12
	get_sensor	12
	have_take	12
	insert	13
	is_configured	13
	is_connected	13
	is_id	13
	is_reading	13
	is_taking	13
	load_configuration	14
	load_location	14
	load_script	14
	load_take	14
	load_take_source	15
	open	15
	reading	15
	read_user_data	15
	replay_take	16
	sample_rate_to_delay	16
	save_configuration	16
	scan	16

set_configuration	16
set_delay	17
set_gain	17
set_gain_sensor	17
set_gselect	18
set_name	18
set_sample_rate	18
set_source	18
set_track_gyroscope_bias	18
start	19
start_take	19
stop	19
stop_take	19
take	19
write_configuration	20
write_user_data	20
3.2 node.usb	20
restore_configuration	20
3.3 node.system	20
calibrate_from_take	22
configuration_matches_take	22
console_client	22
disable_input	23
enable_input	23
export_type	23
file_exists	23
geocode_address	23
get_default	23
get_history	23
get_local_mode	24
get_location_list	24
get_preference	24
get_service	24
get_system_info	24
get_wifi_list	24
get_zeroconf_list	24
initialize	25
is_filename	25
is_initialized	25
load_plugin	25
local_mode	25
location	25
log	26
open_database	26
print	26
quit	26

republish_services	26
restart	26
save_initialization	26
save_location	27
set_date_time	27
set_data_path	27
set_default	27
set_local_mode	27
set_search_path	28
service_exists	28
set_wifi	28
shutdown	28
sleep	28
start_services	29
unique_id	29
unload_plugin	29
3.4 Classes	29
ConfigurationContainer	29
ConfigurationNode	29
LocationNode	30
LocationContainer	30
PreferenceNode	30
PreviewContainer	30
PreviewNode	30
RawContainer	31
RawNode	31
SensorContainer	31
SensorNode	31
ServiceNode	32
SystemInfoNode	32
TakeNode	32
WifiContainer	32
WifiNode	33
ZeroconfContainer	33
ZeroconfNode	33
3.5 Lua	33
boolean	33
iterator	33
nil	33
number	33
string	34
table	34

1 Introduction

The MotionNode system provides a scripting interface based on the Lua programming language. All user interaction with the MotionNode system is implemented through a custom set of Lua commands. This set of commands is defined in a Lua module named `node`.

This document provides an overview of the `node` commands, a detailed description of the embedded Lua interpreter, and a per function reference of the `node` module.

2 Getting Started

2.1 What is Lua?

From the *About* page on the Lua web site:

Lua is a powerful light-weight programming language designed for extending applications. Lua is also frequently used as a general-purpose, stand-alone language. Lua is free software.

Lua is a scripting language. The MotionNode system extends the Lua language with C++ functions and manages a persistent global Lua state. There are multiple input methods for Lua commands, an interactive console, an HTTP remote procedure call interface, an HTTP GET file interpreter, and a regular file interpreter. The MotionNode system accepts input from any source on a first in, first out basis.

This reference only describes Lua with respect to the MotionNode system and the `node` module. For general Lua usage and syntax help please refer to the excellent “Programming in Lua” book. A copy is available online at <http://www.lua.org/pil/>.

2.2 Interactive Console

The MotionNode system implements an interactive Lua console. The console is modeled after the stand alone Lua interpreter. An interactive console session might look something like Example 1. Note that the user input is highlighted.

To start a console, open **Program Files > MotionNode > Run Console (Command Prompt)**. The interactive console is most useful to run advanced commands that are not available through the user interface. For repeated tasks you may want to use script files instead.

```
console> print("Hello World")
Hello World
console> for i=1,5 do
    > print(i)
    > end
1
2
3
4
5
console> EOF (Ctrl-Z on Windows)
```

Example 1: An interactive console session.

2.3 Script File

A script file is simply a text file that contains a sequence of Lua commands. For example, here is a script file that records 10 seconds of motion data and exports the results to a file. The entire script file is parsed into one chunk and executed at once. This has the effect that all of the printed output is displayed after the script has completed.

```
-- Start a take. Sleep for 10 seconds, and then
-- export the data.
if node.start_take() then
    node.system.sleep(10)
    node.close()
    if node.export("take_10sec.fbx") then
        print("Exported take")
    end
end
end

print("Done")
```

Example 2: An example script file.

Script files can be loaded with the `load_script` command, or using the **File > Open** command in the MotionNode User Interface. The HTTP server embedded in the MotionNode system will also handle GET requests for Lua script files. For example, the main MotionNode User Interface is simply the file `/index.lua`. The HTTP server executes the script file and returns the printed results as the file contents.

2.4 Basic Commands

The most basic feature of the MotionNode system is capturing motion data. The `start_take` command will start logging data streams from all Nodes in the current configuration.

```
node.start_take()
```

Example 3: Start a take.

Before running the `start_take` command there must be at least one Node in the configuration. There are many ways to add Nodes to the configuration. Use the `open` command to load a configuration file and replace the current configuration. Use the `scan` command to enumerate available devices and add them to the configuration. Use the `insert` command to manually add a single Node to the configuration. Note that two hyphens (`--`) comment out the rest of the line.

```
-- Open the configuration file "configuration.xml".
-- Replaces current configuration.
node.open("configuration.xml")

-- Scan for any Nodes that are plugged in. Add them
-- to the current configuration.
node.scan()

-- Manually insert a Node entry, adding it to the
-- current configuration.
node.insert("MyNode")
```

Example 4: Commands that add Nodes to the configuration.

The MotionNode system attempts to load a default configuration at start time. First, the default configuration file is loaded if it exists. Second, if the configuration is still empty, the command `scan` is run. See Example 5 for part of the system initialization script. It is also an example of how scripting is used in the MotionNode system.

Most of the `node` commands return a `boolean`, `string` pair. In this case, the `boolean` indicates success or failure and the `string` will contain some description of the result. The result description `string` will be fairly generic. For more detailed error messages refer to the MotionNode system error log.

Many `node` commands take an optional `id` parameter, denoted by a set of

```
-- Load default configuration file.
if node.system.file_exists("default/configuration.xml") then
  node.load_configuration("default/configuration.xml")
end

-- Scan for available devices if the configuration
-- is empty.
if not node.is_configured() then
  node.scan()
end
```

Example 5: Part of system initialization, managing the configuration.

```
-- Repeat the first example, check the result of the
-- command this time.
result, message = node.start_take()
if not result then
  -- Print out the error message.
  print(message)
  -- Or, interrupt execution and return an error.
  --error(message)
end
```

Example 6: An example with error checking.

square brackets ([*id*]). Each configured Node has a unique string identifier, called the *id* field. By specifying an *id* the command operates on a single configured Node. Otherwise, the command operates over all configured Nodes.

3 Module Reference

This section provides a comprehensive list of all functions in the `node` module.

3.1 `node`

```
boolean, string close([string id])
ConfigurationContainer configuration()
boolean, string connect([string id])
ConfigurationContainer connected()
boolean, string define_identity_pose()
number delay_to_sample_rate(number value)
boolean, string erase([string id])
boolean, string export(string filename,
    [string option],
    [string type])
boolean, string export_stream(string filename, [string option])
ConfigurationNode get_node(string key,
    type value,
    [type compare])
ConfigurationNode get_node_by_id(string id)
ConfigurationNode get_node_by_key(number key)
PreviewContainer get_preview()
RawContainer get_raw()
SensorContainer get_sensor()
boolean have_take()
boolean, string insert(string id, [string bus], [string parent_id])
boolean is_configured([string id])
boolean is_connected([string id])
boolean is_id(string id)
boolean is_reading([string id])
boolean is_taking()
boolean, string load_configuration(string filename)
boolean, string load_location(string filename)
boolean, string load_script(string filename)
```

```

boolean, string load_take(string filename)
boolean, string load_take_source([string filename])
boolean, string open(string filename)
ConfigurationContainer reading()
table read_user_data(string id)
boolean, string replay_take([string filename])
number sample_rate_to_delay(number value)
boolean, string save_configuration(string filename)
boolean, string scan([string filter])
boolean, string set_configuration(string key,
    type value,
    [string id])
boolean, string set_delay(number value, [string id])
boolean, string set_gain(number value, [string id])
boolean, string set_gain_sensor(number value, [string id])
boolean, string set_gselect(number value, [string id])
boolean, string set_name(string value, [string id])
boolean, string set_sample_rate(number value, [string id])
boolean, string set_source(string value, [string id])
boolean, string set_track_gyroscope_bias(number value, [string id])
boolean, string start([string id])
boolean, string start_take()
boolean, string stop([string id])
boolean, string stop_take()
TakeNode take()
boolean, string write_configuration(string id,
    [boolean write_factory])
boolean write_user_data(string id, table data)

```

close

```

boolean, string close([string id])
    Precondition id == nil or is_connected(id) == true
                  id ~= nil or connected() ~= nil
    Effect Close existing connection to one or more configured
             Nodes.
    Postcondition is_connected([id]) == false
    Return true iff at least one Node connection was closed.

```

configuration

ConfigurationContainer configuration()

- Precondition** `is_configured() == true`
at least one Node exists in the configuration state
- Postcondition** `configuration():list()` is iterable
- Return** an iterable container of all configured Nodes.

connect

boolean, string connect([string id])

- Precondition** `is_configured([id]) == true`
`is_connected([id]) == false`
- Effect** Connect to one or more configured Nodes.
- Postcondition** `is_connected([id]) == true`
- Return** true iff all requested Node connections are open.

connected

ConfigurationContainer connected()

- Precondition** at least one Node is currently connected
- Postcondition** `connected():list()` is iterable
- Return** an iterable container of all connected Nodes.

define_identity_pose

boolean, string define_identity_pose()

- Precondition** `is_reading() == true`
`is_taking() == false`
- Effect** Define the identity orientation of all configured Nodes.
The local orientation output is defined relative to the identity orientation.
- Return** true iff

delay_to_sample_rate

number delay_to_sample_rate(number value)

- Summary** Convert a unit valued delay number to a sample rate in hertz.
- Return** $(1.625 - \text{value}) * 80$ truncated to 10 hertz increments

erase

boolean, string erase([string id])

Precondition is_configured([id]) == true
is_connected() == false
connected() == nil

Effect Remove one or more configured Nodes from the configuration state.

Postcondition is_configured([id]) == false

Return true iff all requested Nodes are removed from the configuration state.

export

boolean, string export(string filename,
[string option],
[string type])

Precondition system.is_filename(filename) == true
have_take() == true
is_configured() == true
system.configuration_matches_take() == true
type == nil or system.export_type()[type] ~=
nil

Effect Export the current take to an external file format.

Return true iff the current take is successfully exported to the requested file.

export_stream

boolean, string export_stream(string filename, [string option])

Precondition system.is_filename(filename) == true
have_take() == true
is_configured() == true
system.configuration_matches_take() == true

Effect Export data streams from the current take to an external file format.

Return true iff the current take is successfully exported to the requested file.

get_node

ConfigurationNode get_node(string key,
type value,
[*type* compare])

Precondition `is_configured() == true`
Effect Get the configured Node with named property `key == value` .
Return the first Node in the current configuration state with named property `key == value` .

`get_node_by_id`

`ConfigurationNode get_node_by_id(string id)`

Effect `get_node('id', id)`

`get_node_by_key`

`ConfigurationNode get_node_by_key(number key)`

Effect `get_node('key', key)`

`get_preview`

`PreviewContainer get_preview()`

Precondition `is_reading() == true`

Postcondition `get_preview():list()` is iterable

Return an iterable container of Preview data for all configured Nodes.

`get_raw`

`RawContainer get_raw()`

Precondition `is_reading() == true`

Postcondition `get_raw():list()` is iterable

Return an iterable container of Raw data for all configured Nodes.

`get_sensor`

`SensorContainer get_sensor()`

Precondition `is_reading() == true`

Postcondition `get_sensor():list()` is iterable

Return an iterable container of Sensor data for all configured Nodes.

`have_take`

`boolean have_take()`

Precondition `is_taking() == false`

Return true iff a take is currently loaded.

insert

boolean, string insert(string id, [string bus], [string parent_id])

Precondition is_id(id) == true
system.unique_id(id) == true
is_configured(system.unique_id(id)) == false
is_connected() == false
connected() == nil

Effect Insert a new Node into the configuration state. If id already exists in the current configuration, generate a unique identifier.

Postcondition is_configured(system.unique_id(id)) == true

Return true and the new unique identifier iff a new Node is inserted into the configuration state.

is_configured

boolean is_configured([string id])

Precondition id == nil or is_id(id) == true

Return true iff all requested Nodes exist in the configuration state.

is_connected

boolean is_connected([string id])

Precondition is_configured([id]) == true

Return true iff all requested Nodes are currently connected.

is_id

boolean is_id(string id)

Precondition #id > 0

id consists of alphanumeric, underscore (_), and hyphen (-) characters

Return true iff the input string is a valid Node identifier

is_reading

boolean is_reading([string id])

Precondition is_connected([id]) == true

Return true iff all requested Nodes are currently reading data.

is_taking

boolean is_taking()

Precondition is_reading() == true

Return true iff a take is currently in progress.

load_configuration

```
boolean, string load_configuration(string filename)
  Precondition system.file_exists(filename) == true
                 is_connected() == false
                 connected() == nil
  Effect Load a configuration file. Replace the current configuration state. Clear the current take state.
  Postcondition is_configured() == true
                 have_take() == false
  Return true iff the configuration file is successfully read, parsed, and at least one Node is loaded into the configuration state.
```

load_location

```
boolean, string load_location(string filename)
  Precondition system.file_exists(filename) == true
                 is_connected() == false
                 connected() == nil
  Effect Load a location file. Replace the current location state of the system. This defines the geomagnetic field reference.
  Return true iff the location file is successfully loaded and parsed.
```

load_script

```
boolean, string load_script(string filename)
  Precondition system.file_exists(filename) == true
  Effect Load and execute a Lua script file.
  Return true iff the script file is successfully read, parsed, and executed.
```

load_take

```
boolean, string load_take(string filename)
  Precondition system.file_exists(filename) == true
                 is_connected() == false
                 connected() == nil
  Effect Load a take file. Replace the current take state. Replace the current configuration state.
  Postcondition is_configured() == true
                 have_take() == true
  Return true iff the take file is successfully read, parsed, and the associated configuration file is successfully loaded.
```

load_take_source

boolean, string load_take_source([string filename])

Precondition filename == nil or load_take(filename) == true
have_take() == true
is_configured() == true
is_connected() == false
connected() == nil

Effect system.configuration_matches.take() == true
Copy the Raw data source list from the current take into the current configuration source list. Clear the current take state. Optionally pre-load a take to copy the data sources from.

Postcondition have_take() == false

Return true iff the take data files are loaded as the current configuration sources for all configured Nodes.

open

boolean, string open(string filename)

Precondition system.file_exists(filename) == true
is_connected() == false
connected() == nil

Effect Load a configuration, location, Lua script, or take file. Detect the file type and call the appropriate file handler.

Return true iff the file is of a valid type and the file handler successfully loaded the file.

reading

ConfigurationContainer reading()

Postcondition reading():list() is iterable

Return an iterable container of all connected Nodes that are currently streaming data.

read_user_data

table read_user_data(string id)

Precondition is_configured(id) == true
is_connected(id) == false

Return a table of number entries from the user portion of the device memory

replay_take

boolean, string replay_take([string filename])

Precondition load_take_source(filename) == true

Effect Replay a take. Rebuild and run the filtering pipeline for the current take. Optionally pre-load a take to replay.

Postcondition is_taking() == true

Return true iff the the take data sources are successfully copied and a take is in progress, reading from the take data files.

sample_rate_to_delay

number sample_rate_to_delay(number value)

Summary Convert a target sample to a unit valued delay number.

Return $1 - (\text{value} - \text{MIN_RATE}) * 0.0125$

save_configuration

boolean, string save_configuration(string filename)

Precondition system.is_filename(filename) == true
is_configured() == true

Effect Write the current configuration state to a file.

Postcondition system.file_exists(filename) == true

Return true iff the configuration state is successfully written to the requested file.

scan

boolean, string scan([string filter])

Precondition is_connected() == false
connected() == nil

Effect Enumerate all available Nodes not currently configured and insert them into the current configuration state. Optionally limit the devices by class name, for example 'usb' or 'bus'.

Postcondition is_configured() == true

Return true iff at least one Node is inserted into the configuration state.

set_configuration

boolean, string set_configuration(string key,

type value,

[string id])

Precondition `id == nil or is_configured(id) == true`

Effect Set a named member property of one or more configured Nodes. Class member properties in Lua are table entries, indexed by name. For example, `object.property = 1` is equivalent to `object["property"] = 1`. This is a convenience function to set a named property of one or all `ConfigurationNode` objects in the current configuration state.

Return `true` iff the named property key was set to value for at least on Node.

set_delay

boolean, string `set_delay(number value, [string id])`

Precondition `is_connected(id) == false`

Effect `set_configuration('delay', value, id)` . Set the delay parameter of one or all configured Nodes, select the active sample rate. A 0 value maps to the default sample rate, otherwise a smaller value indicates a faster rate.

set_gain

boolean, string `set_gain(number value, [string id])`

Precondition `is_taking() == false`

Effect `set_configuration('gain', value, id)` . Set the gain parameter of one or all configured Nodes, which adjusts the orientation output for specific applications. A value of 0 denotes smooth output, while a value of 1 denotes responsive output.

set_gain_sensor

boolean, string `set_gain_sensor(number value, [string id])`

Precondition `is_taking() == false`

Effect `set_configuration('gain_sensor', value, id)` . Set the sensor gain parameter of one or all configured Nodes, which adjusts post-filtering of the sensor data. A value of 0 denotes smoothest output, while a value of 1 denotes no filtering. Note that this does not effect the orientation output since the filter is applied after the orientation is computed.

set_gselect

boolean, string set_gselect(number value, [string id])

Precondition is_connected(id) == false

Effect set_configuration('gselect', value, id) . Set the gselect parameter of one or all configured Nodes, select the output range of the accelerometer. Any value less than 6 maps to 2 g, otherwise 6 g.

set_name

boolean, string set_name(string value, [string id])

Precondition is_connected(id) == false

Effect set_configuration('name', value, id) . Set the name parameter of one or all configured Nodes.

set_sample_rate

boolean, string set_sample_rate(number value, [string id])

Precondition is_connected(id) == false

Effect set_delay(sample_rate_to_delay(value), id) . Set the delay parameter of one or all configured Nodes by target sample rate.

set_source

boolean, string set_source(string value, [string id])

Summary Set the source parameter of one or all configured Nodes. The source parameter defines the physical input source for a Node. For example, a Node may read from a USB device or from a file on disk.

Precondition is_connected(id) == false

Effect set_configuration('source', value, id) .

set_track_gyroscope_bias

boolean, string set_track_gyroscope_bias(number value, [string id])

Summary Adjust or disable gyroscope bias tracking. User tunable parameter where 1 is maximum tracking and 0 disables tracking.

Precondition is_taking() == false

Effect set_configuration('track_gyroscope_bias', value, id) .

start

boolean, string start([string id])
Precondition is_reading([id]) == false
is_connected([id]) == true or connect([id]) == true
Effect Start reading data from one or more configured Nodes.
Postcondition is_reading([id]) == true
Return true iff all requested Nodes are currently streaming data.

start_take

boolean, string start_take()
Precondition is_taking() == false
is_reading() == true or start() == true
Effect Start a take. Start writing Node data streams to files.
Postcondition is_taking() == true
Return true iff a take is currently in progress.

stop

boolean, string stop([string id])
Precondition is_reading([id]) == true
is_taking() == false or stop_take() == true
Effect Stop reading data from one or more Nodes. Stop the current take.
Postcondition is_reading([id]) == false
Return true iff all requested Nodes stopped streaming data.

stop_take

boolean, string stop_take()
Precondition is_taking() == true
Effect Stop the current take. Save the take definition to a file.
Postcondition is_taking() == false
Return true iff the current take is successfully stopped and all output files are saved.

take

TakeNode take()
Precondition have_take() == true
Return a description of the currently loaded take.

write_configuration

```
boolean, string write_configuration(string id,
    [boolean write_factory])
```

Precondition `is_configured(id) == true`
`is_connected(id) == false`

Effect Write the configuration values for this Node to the on-board non-volatile/flash memory. Not all data sources support onboard memory writes.

write_user_data

```
boolean write_user_data(string id, table data)
```

Precondition `is_configured(id) == true`
`is_connected(id) == false`
`#data > 0 and #data <= 32`

Return true iff the the input data is successfully saved to the user portion of the device memory.

3.2 node.usb

```
boolean, string restore_configuration(string id)
```

restore_configuration

```
boolean, string restore_configuration(string id)
```

Precondition `is_configured(id) == true`
`is_connected(id) == false`

Effect Restore the Node configuration stored in onboard non-volatile memory to the factory setting. The factory setting is simply a backup, and is writable by the command `write_configuration`.

3.3 node.system

```
boolean, string calibrate_from_take(string id,
    [boolean calibrate_accelerometer],
    [boolean calibrate_gyroscope_only])
```

```
boolean, string configuration_matches_take()
```

```
boolean, string console_client(string address, number port)
```

```
boolean, string disable_input()
```

```
boolean, string enable_input(string address, number port)
```

```
table export_type()
```

```
boolean file_exists(string filename)
boolean, string geocode_address(string address)
string get_default(string name)
table get_history()
string get_local_mode()
LocationContainer get_location_list()
PreferenceNode get_preference()
table get_service()
SystemInfoNode get_system_info()
WifiContainer get_wifi_list()
ZeroconfContainer get_zeroconf_list()
boolean, string initialize()
boolean is_filename(string filename)
boolean is_initialized()
boolean, string load_plugin([string name])
table local_mode()
boolean, string location(number latitude,
    number longitude,
    number elevation)
boolean, string log(string filename)
boolean, string open_database(string filename)
void print(type ...)
boolean quit()
boolean republish_services()
boolean restart()
boolean, string save_initialization()
boolean, string save_location([string filename])
boolean set_date_time(string date)
boolean set_data_path(string path)
void set_default(string name, string value)
boolean set_local_mode(string value)
boolean set_search_path(string search_path)
boolean service_exists(string name)
boolean set_wifi(string hwid, string essid, string password)
boolean shutdown(string id)
boolean sleep(number second)
boolean, string start_services(table service)
boolean, string unique_id(string id)
```

boolean, string unload_plugin([string name])

calibrate_from_take

boolean, string calibrate_from_take(string id,

[boolean calibrate_accelerometer],

[boolean calibrate_gyroscope_only])

Precondition have_take() == true
 is_configured(id) == true
 configuration_matches_take() == true

Effect Use the current take to automatically generate calibration values for a Node. This command does not write the calibration values to the device. Use the `write_configuration` command to write the values to the Node onboard non-volatile memory.

`calibrate_accelerometer` Enable accelerometer calibration. Disabled by default since they are factory calibrated and do not require location or mounting based calibration. Note that the function will detect accelerometer only devices and always enable calibration.
`calibrate_gyroscope_only` Enable gyroscope bias calibration mode. Disables all other calibration procedures since they are rotation based. Bias calibration data should be recorded in a static orientation. Does nothing for devices without gyroscopes onboard.

Return true iff the calibration values were successfully generated and copied into the system configuration.

configuration_matches_take

boolean, string configuration_matches_take()

Precondition have_take() == true
 is_configured() == true

Return true iff there is exactly a one to one mapping from configuration entries to take channels.

console_client

boolean, string console_client(string address, number port)

Effect Start an interactive console in the current thread.

disable_input

boolean, string disable_input()

Effect Disable input service. Do not run filtering pipeline.

enable_input

boolean, string enable_input(string address, number port)

Effect Enable remote input service, external filtering pipeline.

export_type

table export_type()

Return an associative array of valid export types.

file_exists

boolean file_exists(string filename)

Return true iff this file exists in the search path

geocode_address

boolean, string geocode_address(string address)

Effect Set the current system location based on an online geocode database lookup.

Parameter address Street address of the current location.

Return true iff the input address was successfully mapped to a valid location.

get_default

string get_default(string name)

Return value of named parameter in the local database system table

get_history

table get_history()

Effect The system keeps track of some commands, for example open, such that the user interface can have some history.

Return an associative array of interactive command parameters.

get_local_mode

string get_local_mode()

Summary Read back the system wide preference for the local rotation coordinate frames.

Effect get_default('local_mode', value)

get_location_list

LocationContainer get_location_list()

Postcondition get_location_list():list() is iterable

Return an iterable container of all previously entered geographic locations

get_preference

PreferenceNode get_preference()

Return the current shared system preferences.

get_service

table get_service()

Return an iterable table of the services loaded at start up.

get_system_info

SystemInfoNode get_system_info()

Return the current system information, an uptime and odometer clock.

get_wifi_list

WifiContainer get_wifi_list()

Return an iterable list of wireless access points in range.

get_zeroconf_list

ZeroconfContainer get_zeroconf_list()

Return an iterable list of active wireless bus devices.

initialize

boolean, string initialize()

Effect Load the initialization script file, which contains all site specific preferences and starts the services.

Return true iff the system is initialized after a call to this function.

is_filename

boolean is_filename(string filename)

Return true iff this filename is a valid name, it is in the search path.

is_initialized

boolean is_initialized()

Return true iff the system has already been initialized.

load_plugin

boolean, string load_plugin([string name])

Precondition is_connected() == false
connected() == nil

Effect Load available plugins. Optionally limit by name.

local_mode

table local_mode()

Return an associative array of local rotation mode constants.

location

boolean, string location(number latitude,
number longitude,
number elevation)

Precondition $-90 < \text{latitude} < 90$
 $-180 < \text{longitude} < 180$
 $-1000 < \text{elevation} < 600000$

Effect Set the current geographic location. Required to estimate the geomagnetic field which varies based on location.

Parameter latitude In decimal degrees.
longitude In decimal degrees.
elevation In meters.

log

boolean, string log(string filename)

Precondition is_filename(filename) == true

Effect Set the current system log filename.

open_database

boolean, string open_database(string filename)

Precondition file_exists(filename) == true

Effect Open the system database file. This is a SQLite version 3 compatible file.

print

void print(*type* ...)

Effect Replacement for the Lua print function. Capture print output such that we can redirect to various output targets.

quit

boolean quit()

Effect Close all services and threads. Exit the program after everything has been shut down.

Return Returns true.

republish_services

boolean republish_services()

Effect Republish the Zeroconf/Bonjour services on the local network. This can be useful if the propagation of the services is taking too much time.

Return Returns true iff services were running and successfully republished.

restart

boolean restart()

Effect Close all services and threads. Re-initialize the system.

Return Returns true.

save_initialization

boolean, string save_initialization()

Effect Save the initialization file based on the current state of the system.

save_location

boolean, string save_location([string filename])

Effect Save a location file based on the current state of the system.

set_date_time

boolean set_date_time(string date)

Summary Set the system clock to the date and time specified in the input parameter. The time is specified in UTC/GMT rather than the local time zone. Format is an ISO string, for example 20101031T103015 for October 31, 2010 at 10:30.15 AM.

Return true iff the clock was successfully updated

set_data_path

boolean set_data_path(string path)

Summary Set the output path for take data as well as the highest precedence entry in the system search path. If the input path name does not exist it will be created.

Precondition is_initialized() == false
path is a valid path name

Postcondition path exists

Parameter path name for take data and other user output files

Return true iff the input path exists and is the current data path

set_default

void set_default(string name, string value)

Effect Set the name value pair in the local database system table.

set_local_mode

boolean set_local_mode(string value)

Summary Sets a system wide preference for the local rotation coordinate frames. Only loaded into a configuration as it is updated. This should precede calls to load files or scan for new Nodes.

Effect set_default('local_mode', value)

Parameter value Select the local rotation mode by string name. The valid parameters are 'WORLD' , 'SENSOR' , and 'WORLD_HEADING' .

set_search_path

boolean set_search_path(string search_path)

Summary Path names that do not exist are not added to the search path. The data path always takes precedence over the search path.

Precondition is_initialized() == false
search_path is a valid list of path names, all of which exist

Postcondition all path names in search_path that exist are in the system search path

Parameter search_path List of semicolon (;) separated path names ordered by precedence

Return Returns true iff all entries in search_path are valid, existing path names

service_exists

boolean service_exists(string name)

Return Returns true iff the named service is implemented in this version of the software.

set_wifi

boolean set_wifi(string hwid, string essid, string password)

Return Returns true iff the requested network is saved as the preferred network. Use empty parameters to clear the preferred network.

shutdown

boolean shutdown(string id)

Effect Connect to a wireless bus by id. Call quit and then shutdown the system if we can. Requires an exclusive lock on the wireless bus.

Return Returns true.

sleep

boolean sleep(number second)

Summary Block the current thread of execution for second seconds. The blocked thread maintains an exclusive lock on the Lua command pipeline.

Parameter second Sleep for this many seconds

Return Always returns true

start_services

boolean, string start_services(table service)

Effect Start named services on requested ports.

unique_id

boolean, string unique_id(string id)

Effect Generate a valid unique Node identifier based on the input string.

unload_plugin

boolean, string unload_plugin([string name])

Precondition is_connected() == false
connected() == nil

Effect Unload currently loaded plugins. Optionally limit by name.

3.4 Classes**ConfigurationContainer**

Simple container for a list of `ConfigurationNode` objects. Use the `list` method as a Lua iterator to access each child node object.

Name	Type	Mutable
list	iterator(ConfigurationNode)	

ConfigurationNode

Name	Type	Mutable
key	number	
id	string	
name	string	true
source	string	true
delay	number	true
gselect	number	true
time_step	number	true
gain	number	true
gain_sensor	number	true
pipeline_select	number	true
node_version	number	
active	boolean	
bus	boolean	
parent	string	true

LocationNode

Simple container for geographic location data.

Name	Type	Mutable
key	number	
name	string	
canonical_address	string	
latitude	number	
longitude	number	
elevation	number	
active	number	

LocationContainer

Name	Type	Mutable
list	iterator(LocationNode)	

PreferenceNode

Name	Type	Mutable
address_input	string	
address_canonical	string	
valid_location	boolean	
latitude	number	
longitude	number	
elevation	number	
data_path	string	
log_path	string	
database_path	string	

PreviewContainer

Name	Type	Mutable
list	iterator(PreviewNode)	

PreviewNode

Simple container for preview data.

Name	Type	Mutable
id	number	
Gqw	number	
Gqx	number	
Gqy	number	
Gqz	number	
Lqw	number	
Lqx	number	
Lqy	number	
Lqz	number	
rx	number	
ry	number	
rz	number	
ax	number	
ay	number	
az	number	

RawContainer

Name	Type	Mutable
list	iterator(RawNode)	

RawNode

Simple container for raw data.

Name	Type	Mutable
id	number	
ax	number	
ay	number	
az	number	
gx	number	
gy	number	
gz	number	
mx	number	
my	number	
mz	number	

SensorContainer

Name	Type	Mutable
list	iterator(SensorNode)	

SensorNode

Simple container for sensor data.

Name	Type	Mutable
id	number	
ax	number	
ay	number	
az	number	
gx	number	
gy	number	
gz	number	
mx	number	
my	number	
mz	number	

ServiceNode

Simple container describes a service.

Name	Type	Mutable
description	string	
port	number	

SystemInfoNode

Simple container for system information. Includes an uptime clock in seconds and an odometer for the total number of samples read.

Name	Type	Mutable
uptime	number	
remaining	number	
connected	number	
odometer	number	
battery_level	number	
battery_state	number	

TakeNode

Simple container for current take information.

Name	Type	Mutable
name	string	
description	string	
start	number	
end	number	
location	string	
loaded_from	string	

WifiContainer

Name	Type	Mutable
list	iterator(WifiNode)	

WifiNode

Simple container for wireless access point information.

Name	Type	Mutable
hwid	string	
ssid	string	
password	string	
type	number	
signal	number	
active	number	

ZeroconfContainer

Name	Type	Mutable
list	iterator(ZeroconfNode)	

ZeroconfNode

Simple container for zeroconf data service information.

Name	Type	Mutable
name	string	
hostname	string	
address	string	
port	number	

3.5 Lua**boolean**

Basic Lua type. See chapter 2.2 in the "Programming in Lua" book for more details (<http://www.lua.org/pil/2.2.html>).

iterator

Lua iterator function. See chapter 4.3.5 in the "Programming in Lua" book for more details (<http://www.lua.org/pil/4.3.5.html>).

nil

Basic Lua type. See chapter 2.1 in the "Programming in Lua" book for more details (<http://www.lua.org/pil/2.1.html>).

number

Basic Lua type. See chapter 2.3 in the "Programming in Lua" book for more details (<http://www.lua.org/pil/2.3.html>).

string

Basic Lua type. See chapter 2.4 in the "Programming in Lua" book for more details (<http://www.lua.org/pil/2.4.html>).

table

Basic Lua type. See chapter 2.5 in the "Programming in Lua" book for more details (<http://www.lua.org/pil/2.5.html>).