Script Definitions (English)

1. Event parameters

1.1 Param - self

Description: self Reference to the event for which the script will be executed. Example -- Output of the event name print(self.name)

1.2 Param - deferredCall

Description: deferredCall Specifies the name of the delay, or nil if the event is not delayed. See defer() function for more information.

2. Global variables/functions

2.1 Keyword - $

Description: $ References an object or event/module by name. In contrast to layout:getEntityByName, which searches an object for a name at runtime, a reference is a fixed shortcut that is uniquely determined at the time of scripting. A reference remains valid even if the name of the referenced object is changed or several objects share the same name. References are comparable to memory addresses and are therefore faster to process than searching for objects. Example -- Referencing of the object with the name "ICE-Lok" local vehicle = $("ICE-Lok") vehicle.currentSpeed = 0

2.2 Keyword - keyword

Description: keyword Represents a value of the type "keyword". A keyword is any object variable from the type "keyword". Keyword variables allow you to classify objects and thus achive a uniform processing, such as triggering an event for all objects that have a certain keyword. Example -- Add the keyword "Locking track" to the object track track.variables["Locking Track"] = keyword -- Check whether the object track has a keyword "Locking track" if track.variables["Locking track"] == keyword then ... end

2.3 Var - layout

Description: layout Global variable to access the properties and contained objects of the layouts. Example local vehicle = layout:getEntityByName("ICE-Lok") local time = layout.time

2.4 Var - network

Description: network Provides access to the networking methods of the external control interface (see Wiki for further information).

2.5 Func - defer

Description: defer (duration, name) Delays the execution of the event by a specified duration. When the function is called, the event is stopped and executed again from the beginning after the duration has elapsed, passing the same parameters as with the first call. Delays can be used to implement time-controlled processes without a timer. Parameters duration - Duration of the delay in seconds. Name - Any name to indicate the delay. The name will be passed as the deferredCall parameter, thus allowing you to distinguish between different delays within an event. Example if not deferredCall then -- First, not delayed call of the event ... -- Delay event by 5 seconds defer(5, "Delay A") elseif deferredCall == "Delay A" then -- Called after 5 seconds ... -- Delay event again defer(23, "Delay B") elseif deferredCall == "Delay B" then -- Called after 23 seconds ... end

2.6 Func - toTime

Description: toTime (x) Converts a value x to a time value between 0 and 24 o'clock. Parameter x - Either a string in the format "HH:MM" or a number between 0 and 1, where 0 is midnight and 0.5 is 12 noon. Return values Returns an object of type Time. Note Time objects support + and - arithmetic and < and > comparison operations if both values are of type time. In addition, time objects are automatically converted into a string in the form "HH:MM" when required. Example -- Set simulation time to 9:45 layout.time = toTime("9:45") -- Increase simulation time by half an hour layout.time = layout.time + toTime("0:30") -- Automatic conversion of a time into a string print("The current simulation time is: " .. layout.time) -- Check whether 12 noon is over if layout.time > toTime(0.5) then

3. Layout

3.1 Prop - activeCameras

Description: activeCameras Represents the active camera of a view, where index 0 represents the main view. In case of cockpit views, the active camera matches the vehicle. Example -- Set main view to vehicle layout.activeCameras[0] = vehicle

3.2 Prop - time

Description: time Represents the current simulation time, as type "Time", see toTime().

3.3 Method - enumEntities

Description: enumEntities (callback) Calls a user-defined function for all objects in the layout, whereby each object is given as a parameter. Example -- Hide all objects layout:enumEntities(function (entity) entity.visible = false end)

3.4 Method - enumVehicles

Description: enumVehicles (callback) Calls a user-defined function for all vehicles in the layout, whereby each vehicle is given as a parameter. Example -- Immediately stop all vehicles layout:enumVehicles(function (vehicle) vehicle.currentSpeed = 0 end)

3.5 Method - getEntityByName

Description: getEntityByName (name) Searches for an object by name. Parameter name - Name of the object to be returned. If several objects in the layout have the same name, the first object found is returned. Return values Returns the object with the specified name or nil if no object could be found. Note The keyword $ is recommended for constant referencing of objects.

3.6 Method - getEntitiesByName

Description: getEntitiesByName (name) Searches for all objects by name. Parameter name - Name of the objects to be returned. Return values Returns a list of all objects with the specified name.

3.7 Method - getEntitiesByKeyword

Description: getEntitiesByKeyword (name) Searches for all objects using a keyword. Parameter name - Name of the keyword to search for. Return values Returns a list of all objects with the specified keyword.

3.8 Method - getEntitiesLinkedTo

Description: getEntitiesLinkedTo (target) Returns all objects which are directly linked to a given target object. Parameter target - Reference to target object. Return values Returns a list of all objects linked to the target.

3.9 Method - getRoutesByName

Description: getRoutesByName (name) Searches for all routes with a specific name. Parameter name - Name to search for. Return values Returns a list of all routes with the specified name.

3.10 Method - getRoutesByKeyword

Description: getRoutesByKeyword (name) Searches for all routes using a keyword. Parameter name - Name of the keyword to search for. Return values Returns a list of all routes with the specified keyword.

3.11 Method - getVehicleGroup

Description: getVehicleGroup (vehicle, [filter]) Determines all vehicles that belong to a vehicle group. Parameter vehicle - Vehicle for which the vehicle group is to be determined. filter - Optional number to return only a specific type of vehicle: 0 - All vehicles 1 - Vehicles with an engine 2 - Vehicles with an active engine 3 - Vehicles without an engine Return values Value 1: A list with all vehicles of the vehicle group, starting with the foremost vehicle (relative to the driving direction of the vehicle group). Value 2: A

list with the orientation of each vehicle, relative to the driving direction of the vehicle group (1 = same direction, -1 = opposite direction).

3.12 Method - getVehicleGroupLength

Description: getVehicleGroupLength (vehicle) Determines the total length of a vehicle group. Parameter vehicle - Vehicle whose total length is to be determined. Return values Returns the vehicle group length in m.

3.13 Method - getVehiclesOn

Description: getVehiclesOn (object) Determines all vehicles that are positioned on a specific track or track contact. Parameter object - Track or track contact to check. Return values Returns a list of all vehicles on the track/track contact.

3.14 Method - getVehiclesOnRoute

Description: getVehiclesOnRoute (contact0, contact1) Finds all vehicles that are on the shortest route between two track contacts. Parameter contact0 - Track contact that marks the start of the route. contact1 - Track contact that marks the end of the route. Return values Returns a list of all vehicles on the specified route. Note The distance between both track contacts must not exceed 150 m. The alignment and activation status of the track contacts have no influence on the route; the shortest way between both contacts is checked.

3.15 Method - getEventsByName

Description: getEventsByName (name) Searches for all events/modules with a specific name. Parameter name - Name to search for. Return values Returns a list of all events/modules with the specified name.

4. Objects

4.1 Prop - name

Description: name Represents the name of an object.

4.2 Prop - visible

Description: visible Controls the visibility of an object (Boolean).

4.3 Prop - link

Description: link Contains a reference to a target object to which the object is linked, or nil if no link exists.

4.4 Prop - trackContact

Description: trackContact Allows access to the track contact functions of an object. Is nil when the object is not a track contact.

4.5 Prop - crane

Description: crane Allows access to the crane functions of an object. Is nil when the object is not a crane.

4.6 Prop - size

Description: size Contains the size of an object as a table with the fields x, y and z (3D vector).

4.7 Prop - transformation

Description: transformation Gives access to the transformation (position, rotation, scaling) of an object. All units are interpreted in meters and scale 1:1.

4.8 Prop - variables

Description: variables Allows access to the variables of an object. Example -- Add/Edit variable track.variables["Counter"] = 1 -- Delete variable track.variables["Counter"] = nil

4.9 Prop - actions

Description: actions Gives access to the switch actions of an object. The index corresponds to the switch name. If an empty index is passed, all switches are affected. Example -- Activate a switch with the name "Light" object.actions["Light"].state = 1 -- Deactivate all switches object.actions[""].state = 0

4.10 Prop - animations

Description: animations Gives access to the animation functions of an object. The index corresponds to the animation name. If an empty index is passed, all animations are affected. Example -- Play an animation with the name "Pantograph" vehicle.animations["Pantograph"]:play() -- Stop all animations vehicle.animations[""]:stop()

4.11 Prop - sounds

Description: sounds Gives access to the sound functions of a sound source. The index corresponds to the sound name. If an empty index is passed, all sounds are affected. Example -- Play a sound called "Horn" vehicle.sounds["Horn"]:play() -- Stop all sounds vehicle.sounds[""]:stop()

4.12 Prop - labels

Description: labels Gives access to the labels of an object. The index corresponds to the label name. Example -- Set a label with name "Target" object.labels["Target"].text = "Berlin"

4.13 Prop - mute

Description: mute Enables/Disables all object sounds.

4.14 Prop - volume

Description: volume Defines the volume of all sounds between 0 (mute) and 100 (maximum volume).

5. Objects (transformation)

5.1 Prop - position

Description: position Contains the position of an object as a table with the fields x, y and z (3D vector).

5.2 Prop - rotation

Description: rotation Contains the rotation of an object as a table with the fields x, y, z and w (Quaternion).

5.3 Prop - scaling

Description: scaling Contains the scaling of an object as a floating point number.

5.4 Method - reset

Description: reset () Resets the position, rotation and scaling to the default values.

5.5 Method - resetPosition

Description: resetPosition () Resets the position to the default value.

5.6 Method - resetRotation

Description: resetRotation () Resets the rotation to the default value.

5.7 Method - resetScaling

Description: resetScaling () Resets the scaling to the default value.

5.8 Method - translate

Description: translate (x, y, z) Moves the object along the coordinate axes x, y and z.

5.9 Method - translateX

Description: translateX (x) Moves the object along the coordinate axis x.

5.10 Method - translateY

Description: translateY (y) Moves the object along the coordinate axis y.

5.11 Method - translateZ

Description: translateZ (z) Moves the object along the coordinate axis z.

5.12 Method - rotate

Description: rotate (x, y, z, w) Rotates the object around the quaternion (x, y, z, w).

5.13 Method - rotateX

Description: rotateX (r) Rotates the object around the coordinate axis x with the angle r (radian).

5.14 Method - rotateY

Description: rotateY (r) Rotates the object around the coordinate axis y with the angle r (radian).

5.15 Method - rotateZ

Description: rotateZ (r) Rotates the object around the coordinate axis z with the angle r (radian).

6. Tracks and streets

6.1 Prop - locked

Description: locked Locks a turnout so that it can not be modified anymore (Boolean).

6.2 Prop - stateCount

Description: stateCount Returns the number of turnout positions.

6.3 Prop - state

Description: state Sets the active turnout position.

6.4 Prop - routeCount

Description: routeCount Returns the number of routes.

6.5 Prop - routes

Description: routes Allows access to individual routes by index, starting at 0. Example -- Activate first route of the track track.routes[0].active = true

7. Tracks and streets (routes)

7.1 Prop - active

Description: active Contains the activation state of a route (Boolean).

7.2 Prop - length

Description: length Contains the length of the route in m.

8. Track contacts

8.1 Prop - connection

Description: connection Allows access to the signal, track or switch to which the track contact is connected to (read only).

8.2 Prop - autoAcceleration

Description: autoAcceleration Enables or disables the automatic speed control (acceleration).

8.3 Prop - autoAccelerationSpeed

Description: autoAccelerationSpeed Contains the target speed in km/h to which a vehicle is accelerated to when speed control is enabled.

8.4 Prop - autoDeceleration

Description: autoDeceleration Enables or disables the automatic speed control (slow down).

8.5 Prop - autoDecelerationSpeed

Description: autoDecelerationSpeed Contains the target speed in km/h to which a vehicle is slowed down when speed control is enabled.

8.6 Prop - uncoupling

Description: uncoupling Enables or disables the uncoupling feature of the track contact.

9. Signals

9.1 Prop - stateCount

Description: stateCount Returns the number of signal terms.

9.2 Prop - state

Description: state Sets the active signal term.

9.3 Prop - connection

Description: connection Contains a reference to the object to which the signal is connected to, or nil if no connection exists.

10. Switches/Controllers

10.1 Prop - stateCount

Description: stateCount Returns the number of switch states.

10.2 Prop - state

Description: state Sets the active switch state.

10.3 Prop - value

Description: value Sets the controller value (between 0 and 1).

10.4 Prop - connection

Description: connection Contains a reference to a target object which is controlled by the switch/controller, or nil if no connection exists.

11. Switches (integrated)

11.1 Prop - stateCount

Description: stateCount Returns the number of switch states.

11.2 Prop - state

Description: state Sets the active switch state.

11.3 Prop - autoMode

Description: autoMode Controls the automatic mode of the switch. Values nil - Disables the automatic mode "Day" - Enables the automatic mode "Switch on at day" "Night" - Enables the automatic mode "Switch on at night"

11.4 Method - nextState

Description: nextState () Activates the next switch state.

12. Vehicles

12.1 Prop - maxSpeed

Description: maxSpeed Contains the maximum speed of the vehicle in km/h.

12.2 Prop - currentSpeed

Description: currentSpeed Contains the current speed of the vehicle in km/h.

12.3 Prop - currentSpeedAbs

Description: currentSpeedAbs Contains the current speed of the vehicle in km/h, relative to the driving direction. A negative value flips the driving direction.

12.4 Prop - targetSpeed

Description: targetSpeed Contains the target speed of the vehicle in km/h.

12.5 Prop - targetSpeedAbs

Description: targetSpeedAbs Contains the target speed of the vehicle in km/h, relative to the driving direction. A negative value flips the driving direction.

12.6 Prop - drivingDirection

Description: drivingDirection Contains the current driving direction of the vehicle (1 for forward, -1 for backward). For stopped vehicles, the last driving direction is returned, or the direction in which the vehicle will go for the next acceleration.

12.7 Prop - acceleration

Description: acceleration Contains the vehicle's positive acceleration in m/s².

12.8 Prop - deceleration

Description: deceleration Contains the vehicle's negative acceleration (braking) in m/s².

12.9 Prop - autoAcceleration

Description: autoAcceleration Enables or disables the automatic acceleration.

12.10 Prop - autoDeceleration

Description: autoDeceleration Enables or disables the automatic slow down / soft coupling.

12.11 Prop - parkingBrake

Description: parkingBrake Enables or disables the parking brake of a non-powered vehicle.

12.12 Prop - particlesEnabled

Description: particlesEnabled Enables or disables the steam particles.

12.13 Prop - target

Description: target Specifies the destination of the vehicle. Depending on the destination, the vehicle automatically selects the shortest path leading to the specified destination when making turns. The destination is either a single track contact, a list of track contacts to be driven to in sequence, or nil to delete a route.

12.14 Prop - length

Description: length Contains the length of the vehicle (distance between both couplers) in m.

12.15 Prop - engine

Description: engine Allows access to the engine of a vehicle, if it exists. Example -- Start engine vehicle.engine.active = true

12.16 Prop - couplers

Description: couplers Allows access to the vehicle's couplers with index 0 (front coupler) and index 1 (rear coupler). Example -- Deactivate rear coupler vehicle.couplers[1].enabled = false

12.17 Method - isLocatedOn

Description: isLocatedOn (object) Checks whether a vehicle is positioned on a track/street or a track

contact. Parameter object - Track/street or track contact to be checked for. Return values Returns true if the vehicle is positioned on the track/street or the track contact, otherwise false.

12.18 Method - isHeadingToward

Description: isHeadingToward (object) Checks whether a vehicle is heading toward a track/street or a track contact. In case of a train, the check is performed starting with the first vehicle in driving direction. Parameter object - Track/street or track contact to be checked for. Return values Returns true if the vehicle is heading toward the track/street or the track contact, otherwise false.

12.19 Method - hasEngine

Description: hasEngine () Indicates whether the vehicle is a powered rolling stock or a wagon. Return values Returns true if the vehicle has its own "engine", otherwise false.

13. Vehicles (engine)

13.1 Prop - active

Description: active Defines if an existing engine is powered on or off.

14. Vehicles (couplers)

14.1 Prop - vehicle

Description: vehicle Contains a reference to the vehicle which owns the coupler.

14.2 Prop - enabled

Description: enabled Contains the activation state of a coupler.

14.3 Prop - connectedCoupler

Description: connectedCoupler Contains a reference to the coupler of a connected vehicle or nil, if the coupler is disconnected.

15. Virtual depots

15.1 Prop - count

Description: count Returns the number of trains/vehicles in the depot.

15.2 Prop - entries

Description: entries Allows access to individual depot entries by index, starting at 0. Example -- Get the name of the first train in the depot name = depot.entries[0].name

15.3 Method - add

Description: add (vehicle) Adds a train/vehicle to the depot. Parameter vehicle - Vehicle to add to the depot. If the vehicle is part of a train, then the whole train is added to the depot.

15.4 Method - release

Description: release (index, [target]) Releases a train/vehicle from the depot. Parameter index - Defines the index of the train to release. The first train in the depot has the index 0. target - Optional, defines a virtual depot to release the train at.

15.5 Method - getEntriesByName

Description: getEntriesByName (name) Determines all depot entries by name. Parameter name - Name of the entries to be returned. Return values Returns a list of all indexes of entries with the specified name.

15.6 Method - getEntriesByKeyword

Description: getEntriesByKeyword (name) Determines all depot entries by keyword. Parameter name - Name of the keyword. Return values Returns a list of all indexes of entries with the specified keyword.

16. Virtual depots (entries)

16.1 Prop - name

Description: name Returns the name of the train.

16.2 Prop - speed

Description: speed Returns the speed of the train when leaving the depot.

16.3 Prop - variables

Description: variables Allows access to the variables of the train (traction vehicle). Example -- Add/Edit variable depot.entries[0].variables["Counter"] = 1 -- Delete variable depot.entries[0].variables["Counter"] = nil

17. Portals

17.1 Prop - connection

Description: connection Contains a reference to the connected portal, or nil if no connection exists.

18. Cranes

18.1 Method - moveTo

Description: moveTo (target) Moves a crane to a new destination. Parameter target - Target object to which the crane is to move. If the crane is not yet moving an object, the crane moves to the target and picks it up. If an object is carried, the crane puts the object onto the target. If target = nil, the connection to the current object is removed.

18.2 Method - reset

Description: reset () Removes the connection to a currently carried object and returns to the starting position.

18.3 Prop - goods

Description: goods Returns the goods that the crane is currently transporting, or nil if there is no transport.

19. Cameras

19.1 Prop - target

Description: target Contains the object what the camera is currently tracking, or nil if no tracking is active.

19.2 Prop - fov

Description: fov Specifies the field of view of the camera, in degrees.

20. Animations

20.1 Prop - position

Description: position Returns the current position of the animation between the beginning (0) and end (1).

20.2 Method - play

Description: play ([position, [direction, [min, [max, [speed]]]]]) Plays the animation. Parameter position - Plays the animation from the beginning (0) or end (1). The default value is -1, in this case the animation will be played from the current position. direction - Plays the animation forward (1) or backward (-1). The

default value is 0, in this case the animation will be played with the current direction. min - Defines the minimum position or -1, if the lower limit should not be changed. An animation is always only played between the lower and upper limit. max - Defines the maximum position or -1, if the upper limit should not be changed. An animation is always only played between the lower and upper limit. speed - Defines the speed or 0, if the current animation speed should not be changed.

20.3 Method - stop

Description: stop ([position]) Stops the animation. Parameter position - Stops the animation at a specific position (between 0 = begining and 1 = end). The default value is -1, in this case the animation is stopped at the current position.

21. Sounds

21.1 Method - play

Description: play () Plays the sound.

21.2 Method - stop

Description: stop () Stops the sound.

22. Labels

22.1 Prop - text

Description: text Contains the text of the label.

22.2 Prop - textAsNumber

Description: textAsNumber Contains the text of the label, interpreted as a number. Note textAsNumber must be used if the label contains numbers and should be used for calculation, since Lua makes a strict distinction between strings and numbers.

23. Text boxes

23.1 Prop - text

Description: text Contains the text of the text box.

23.2 Prop - textAsNumber

Description: textAsNumber Contains the text of the text box, interpreted as a number. Note textAsNumber must be used if the text box contains numbers and should be used for calculation, since

Lua makes a strict distinction between strings and numbers.

24. Particle effects

24.1 Prop - active

Description: active Contains the activation state of the dynamic particle effect.

25. Routes

25.1 Prop - name

Description: name Specifies the name of the route.

25.2 Prop - active

Description: active Defines if the route is active or inactive.

25.3 Prop - state

Description: state Contains the detailed state of the route. Possible values 0 - Route is inactive. 1 - Route is requested, but is currently blocked by another route. 2 - Route is about to be activated but waits for turnouts/signals to reach their defined states. 3 - Route is active and locked.

25.4 Prop - autoActivate

Description: autoActivate Specifies whether a route should be automatically requested again as soon as it is released. This property is suitable for central blocks, where a route that is still active is already requested by another train. The request is automatically processed as soon as the previous train releases the route.

25.5 Prop - waypoints

Description: waypoints Allows access to the waypoints of a route as a list of track contacts.

25.6 Prop - tracks

Description: tracks Allows access to the tracks of a route as a list.

25.7 Prop - variables

Description: variables Allows access to the variables of a route. Example -- Add/Edit variable route.variables["Counter"] = 1 -- Delete variable route.variables["Counter"] = nil

25.8 Method - canActivate

Description: canActivate () Checks whether the route is free and can be activated immediately without blocking (true).

26. Time values

26.1 Prop - value

Description: value Returns the time value as a number between 0 and 1.

26.2 Prop - hour

Description: hour Returns the hour of a time value.

26.3 Prop - min

Description: min Returns the minute of a time value.

27. Events/Modules

27.1 Prop - name

Description: name Contains the name of the event/module.

27.2 Prop - enabled

Description: enabled Contains the activation state of the event/module.

28. Modules

28.1 Prop - timers

Description: timers Allows access to the timers of an event module. The index corresponds to the timer name. Example -- Start a timer with the name "Clock generator" module.timers["Clock generator"]:start(2, true)

28.2 Prop - variables

Description: variables Allows access to the variables of an event module. Example -- Add/Edit variable module.variables["Counter"] = 1 -- Delete variable module.variables["Counter"] = nil

29. Events (user-defined)

29.1 Method - invoke

Description: invoke (...) Triggers the user-defined event with the passed parameters. Example local event = $("Custom event") event:invoke(42, "Text")

30. Timers

30.1 Prop - active

Description: active Indicates whether the timer is still running or has already expired.

30.2 Method - start

Description: start (duration, [restart]) Starts the timer from the beginning. Parameter duration - Duration in seconds. restart - Specifies whether the timer should restart automatically (true) or not (false). The default value is true.

30.3 Method - stop

Description: stop () Stops the timer.

31. Network

31.1 Method - notify

Description: notify (name, [params]) Sends a notification to all connected network clients with the help of the external control interface. Parameter Name - Custom notification name. Params - Optional parameters to send with.

32. Lua base functions

32.1 Var - _G

Description: _G A global variable that holds the global environment.

32.2 Var - _VERSION

Description: _VERSION A global variable that holds a string containing the running Lua version. The current value of this variable is "Lua 5.3".

32.3 Func - assert

Description: assert (v, [message]) Calls error if the value of its argument v is false (i.e., nil or false); otherwise, returns all its arguments. In case of error, message is the error object; when absent, it defaults

to "assertion failed!"

32.4 Func - error

Description: error (message, [level]) Terminates the last protected function called and returns message as the error object. Function error never returns. Usually, error adds some information about the error position at the beginning of the message, if the message is a string. The level argument specifies how to get the error position. With level 1 (the default), the error position is where the error function was called. Level 2 points the error to where the function that called error was called; and so on. Passing a level 0 avoids the addition of error position information to the message.

32.5 Func - ipairs

Description: ipairs (t) Returns three values (an iterator function, the table t, and 0) so that the construction for i,v in ipairs(t) do body end will iterate over the key–value pairs (1,t[1]), (2,t[2]), ..., up to the first nil value.

32.6 Func - next

Description: next (table, [index]) Allows a program to traverse all fields of a table. Its first argument is a table and its second argument is an index in this table. next returns the next index of the table and its associated value. When called with nil as its second argument, next returns an initial index and its associated value. When called with the last index, or with nil in an empty table, next returns nil. If the second argument is absent, then it is interpreted as nil. In particular, you can use next(t) to check whether a table is empty. The order in which the indices are enumerated is not specified, even for numeric indices. (To traverse a table in numerical order, use a numerical for.) The behavior of next is undefined if, during the traversal, you assign any value to a non-existent field in the table. You may however modify existing fields. In particular, you may clear existing fields.

32.7 Func - pairs

Description: pairs (t) If t has a metamethod __pairs, calls it with t as argument and returns the first three results from the call. Otherwise, returns three values: the next function, the table t, and nil, so that the construction for k,v in pairs(t) do body end will iterate over all key–value pairs of table t. See function next for the caveats of modifying the table during its traversal.

32.8 Func - pcall

Description: pcall (f, arg1, ...) Calls function f with the given arguments in protected mode. This means that any error inside f is not propagated; instead, pcall catches the error and returns a status code. Its first result is the status code (a boolean), which is true if the call succeeds without errors. In such case, pcall also returns all results from the call, after this first result. In case of any error, pcall returns false plus the error message.

32.9 Func - print

Description: print (...) Receives any number of arguments and prints their values to the event protocol, using the tostring function to convert each argument to a string. print is not intended for formatted output, but only as a quick way to show a value, for instance for debugging. For complete control over the output, use string.format and io.write.

32.10 Func - select

Description: select (index, ...) If index is a number, returns all arguments after argument number index; a negative number indexes from the end (-1 is the last argument). Otherwise, index must be the string "#", and select returns the total number of extra arguments it received.

32.11 Func - tonumber

Description: tonumber (e, [base]) When called with no base, tonumber tries to convert its argument to a number. If the argument is already a number or a string convertible to a number, then tonumber returns this number; otherwise, it returns nil. The conversion of strings can result in integers or floats, according to the lexical conventions of Lua. (The string may have leading and trailing spaces and a sign.) When called with base, then e must be a string to be interpreted as an integer numeral in that base. The base may be any integer between 2 and 36, inclusive. In bases above 10, the letter 'A' (in either upper or lower case) represents 10, 'B' represents 11, and so forth, with 'Z' representing 35. If the string e is not a valid numeral in the given base, the function returns nil.

32.12 Func - tostring

Description: tostring (e) Receives a value of any type and converts it to a string in a human-readable format. (For complete control of how numbers are converted, use string.format.)

32.13 Func - type

Description: type (v) Returns the type of its only argument, coded as a string. The possible results of this function are "nil" (a string, not the value nil), "number", "string", "boolean", "table", "object", "function", "thread", and "userdata".

32.14 Func - xpcall

Description: xpcall (f, msgh, [arg1, ...]) This function is similar to pcall, except that it sets a new message handler msgh.

33. Lua libraries

33.1 Module - math

Description: math This library provides basic mathematical functions. It provides all its functions and constants inside the table math.

33.2 Module - string

Description: string This library provides generic functions for string manipulation, such as finding and extracting substrings, and pattern matching. When indexing a string in Lua, the first character is at position 1 (not at 0, as in C). Indices are allowed to be negative and are interpreted as indexing backwards, from the end of the string. Thus, the last character is at position -1, and so on. The string library provides all its functions inside the table string. It also sets a metatable for strings where the __index field points to the string table. Therefore, you can use the string functions in object-oriented style. For instance, string.byte(s,i) can be written as s:byte(i). The string library assumes one-byte character encodings.

33.3 Module - utf8

Description: utf8 This library provides basic support for UTF-8 encoding. It provides all its functions inside the table utf8. This library does not provide any support for Unicode other than the handling of the encoding. Any operation that needs the meaning of a character, such as character classification, is outside its scope. Unless stated otherwise, all functions that expect a byte position as a parameter assume that the given position is either the start of a byte sequence or one plus the length of the subject string. As in the string library, negative indices count from the end of the string.

33.4 Module - table

Description: table This library provides generic functions for table manipulation. It provides all its functions inside the table table.

34. Lua math functions

34.1 Class var - huge

Description: huge The float value HUGE_VAL, a value larger than any other numeric value.

34.2 Class var - maxinteger

Description: maxinteger An integer with the maximum value for an integer.

34.3 Class var - mininteger

Description: mininteger An integer with the minimum value for an integer.

34.4 Class var - pi

Description: pi The value of Pi.

34.5 Class func - abs

Description: abs (x) Returns the absolute value of x.

34.6 Class func - acos

Description: acos (x) Returns the arc cosine of x (in radians).

34.7 Class func - asin

Description: asin (x) Returns the arc sine of x (in radians).

34.8 Class func - atan

Description: atan (y, [x]) Returns the arc tangent of y/x (in radians), but uses the signs of both arguments to find the quadrant of the result. (It also handles correctly the case of x being zero.) The default value for x is 1, so that the call math.atan(y) returns the arc tangent of y.

34.9 Class func - ceil

Description: ceil (x) Returns the smallest integral value larger than or equal to x.

34.10 Class func - cos

Description: cos (x) Returns the cosine of x (assumed to be in radians).

34.11 Class func - deg

Description: deg (x) Converts the angle x from radians to degrees.

34.12 Class func - exp

Description: exp (x) Returns the value e^x (where e is the base of natural logarithms).

34.13 Class func - floor

Description: floor (x) Returns the largest integral value smaller than or equal to x.

34.14 Class func - fmod

Description: fmod (x, y) Returns the remainder of the division of x by y that rounds the quotient towards zero.

34.15 Class func - log

Description: log(x, [base]) Returns the logarithm of x in the given base. The default for base is e (so that the function returns the natural logarithm of x).

34.16 Class func - max

Description: max (x, ...) Returns the argument with the maximum value.

34.17 Class func - min

Description: min (x, ...) Returns the argument with the minimum value.

34.18 Class func - modf

Description: modf (x) Returns the integral part of x and the fractional part of x. Its second result is always a float.

34.19 Class func - rad

Description: rad (x) Converts the angle x from degrees to radians.

34.20 Class func - random

Description: random ([m, [n]]) When called without arguments, returns a pseudo-random float with uniform distribution in the range [0,1). When called with two integers m and n, math.random returns a pseudo-random integer with uniform distribution in the range [m, n]. (The value n-m cannot be negative and must fit in a Lua integer.) The call math.random(n) is equivalent to math.random(1,n).

34.21 Class func - randomseed

Description: randomseed (x) Sets x as the "seed" for the pseudo-random generator: equal seeds produce equal sequences of numbers.

34.22 Class func - sin

Description: sin (x) Returns the sine of x (assumed to be in radians).

34.23 Class func - sqrt

Description: sqrt (x) Returns the square root of x. (You can also use the expression x^0.5 to compute this value.)

34.24 Class func - tan

Description: tan (x) Returns the tangent of x (assumed to be in radians).

34.25 Class func - tointeger

Description: tointeger (x) If the value x is convertible to an integer, returns that integer. Otherwise, returns nil.

34.26 Class func - type

Description: type (x) Returns "integer" if x is an integer, "float" if it is a float, or nil if x is not a number.

34.27 Class func - ult

Description: ult (m, n) Returns a boolean, true if and only if integer m is below integer n when they are compared as unsigned integers.

35. Lua string functions

35.1 Class func - byte

Description: byte (s, [i, [j]]) Returns the internal numeric codes of the characters s[i], s[i+1], ..., s[j]. The default value for i is 1; the default value for j is i. These indices are corrected following the same rules of function string.sub.

35.2 Class func - char

Description: char (...) Receives zero or more integers. Returns a string with length equal to the number of arguments, in which each character has the internal numeric code equal to its corresponding argument.

35.3 Class func - find

Description: find (s, pattern, [init, [plain]])) Looks for the first match of pattern in the string s. If it finds a match, then find returns the indices of s where this occurrence starts and ends; otherwise, it returns nil. A third, optional numeric argument init specifies where to start the search; its default value is 1 and can be negative. A value of true as a fourth, optional argument plain turns off the pattern matching facilities, so the function does a plain "find substring" operation, with no characters in pattern being considered magic. Note that if plain is given, then init must be given as well. If the pattern has captures, then in a successful match the captured values are also returned, after the two indices.

35.4 Class func - format

Description: format (formatstring, ...) Returns a formatted version of its variable number of arguments following the description given in its first argument (which must be a string). The format string follows the same rules as the ISO C function sprintf. The only differences are that the options/modifiers *, h, L, l, n,

and p are not supported and that there is an extra option, q. The q option formats a string between double quotes, using escape sequences when necessary to ensure that it can safely be read back by the Lua interpreter. For instance, the call string.format('%q', 'a string with "quotes" and \n new line') may produce the string: "a string with \"quotes\" and \new line" Options A, a, E, e, f, G, and g all expect a number as argument. Options c, d, i, o, u, X, and x expect an integer. When Lua is compiled with a C89 compiler, options A and a (hexadecimal floats) do not support any modifier (flags, width, length). Option s expects a string; if its argument is not a string, it is converted to one following the same rules of tostring. If the option has any modifier (flags, width, length), the string argument should not contain embedded zeros.

35.5 Class func - gmatch

Description: gmatch (s, pattern) Returns an iterator function that, each time it is called, returns the next captures from pattern over the string s. If pattern specifies no captures, then the whole match is produced in each call. As an example, the following loop will iterate over all the words from string s, printing one per line: s = "hello world from Lua" for w in string.gmatch(s, "%a+") do print(w) end The next example collects all pairs key=value from the given string into a table: t = {} s = "from=world, to=Lua" for k, v in string.gmatch(s, "(%w+)=(%w+)") do t[k] = v end For this function, a caret '^' at the start of a pattern does not work as an anchor, as this would prevent the iteration.

35.6 Class func - gsub

Description: gsub (s, pattern, repl, [n]) Returns a copy of s in which all (or the first n, if given) occurrences of the pattern have been replaced by a replacement string specified by repl, which can be a string, a table, or a function. gsub also returns, as its second value, the total number of matches that occurred. The name gsub comes from Global SUBstitution. If repl is a string, then its value is used for replacement. The character % works as an escape character: any sequence in repl of the form %d, with d between 1 and 9, stands for the value of the d-th captured substring. The sequence %0 stands for the whole match. The sequence %% stands for a single %. If repl is a table, then the table is queried for every match, using the first capture as the key. If repl is a function, then this function is called every time a match occurs, with all captured substrings passed as arguments, in order. In any case, if the pattern specifies no captures, then it behaves as if the whole pattern was inside a capture. If the value returned by the table query or by the function call is a string or a number, then it is used as the replacement string; otherwise, if it is false or nil, then there is no replacement (that is, the original match is kept in the string). Here are some examples: x = string.gsub("hello world", "(%w+)", "%1 %1") --> x="hello hello world world" x = string.gsub("hello world", "%w+", "%0 %0", 1) --> x="hello hello world" x = string.gsub("hello world from Lua", "(%w+)%s*(%w+)", "%2 %1") --> x="world hello Lua from" x = string.gsub("home = $HOME, user = $USER", "%$(%w+)", os.getenv) --> x="home = /home/roberto, user = roberto" x = string.gsub("4+5 = $return 4+5$", "%$(.-)%$", function (s) return load(s)() end) --> x="4+5 = 9" local t = {name="lua", version="5.3"} x = string.gsub("$name-$version.tar.gz", "%$(%w+)", t) --> x="lua-5.3.tar.gz"

35.7 Class func - len

Description: len (s) Receives a string and returns its length. The empty string "" has length 0. Embedded zeros are counted, so "a\000bc\000" has length 5.

35.8 Class func - lower

Description: lower (s) Receives a string and returns a copy of this string with all uppercase letters changed to lowercase. All other characters are left unchanged. The definition of what an uppercase letter is depends on the current locale.

35.9 Class func - match

Description: match (s, pattern, [init]) Looks for the first match of pattern in the string s. If it finds one, then match returns the captures from the pattern; otherwise it returns nil. If pattern specifies no captures, then the whole match is returned. A third, optional numeric argument init specifies where to start the search; its default value is 1 and can be negative.

35.10 Class func - rep

Description: rep (s, n) Returns a string that is the concatenation of n copies of the string s separated by the string sep. The default value for sep is the empty string (that is, no separator). Returns the empty string if n is not positive.

35.11 Class func - reverse

Description: reverse (s) Returns a string that is the string s reversed.

35.12 Class func - sub

Description: sub (s, i, [j]) Returns the substring of s that starts at i and continues until j; i and j can be negative. If j is absent, then it is assumed to be equal to -1 (which is the same as the string length). In particular, the call string.sub(s,1,j) returns a prefix of s with length j, and string.sub(s, -i) (for a positive i) returns a suffix of s with length i. If, after the translation of negative indices, i is less than 1, it is corrected to 1. If j is greater than the string length, it is corrected to that length. If, after these corrections, i is greater than j, the function returns the empty string.

35.13 Class func - upper

Description: upper (s) Receives a string and returns a copy of this string with all lowercase letters changed to uppercase. All other characters are left unchanged. The definition of what a lowercase letter is depends on the current locale.

36. Lua utf8 functions

36.1 Class func - char

Description: char (...) Receives zero or more integers, converts each one to its corresponding UTF-8 byte sequence and returns a string with the concatenation of all these sequences.

36.2 Class var - charpattern

Description: charpattern The pattern (a string, not a function) "[\0-\x7F\xC2-\xF4][\x80-\xBF]*", which matches exactly one UTF-8 byte sequence, assuming that the subject is a valid UTF-8 string.

36.3 Class func - codes

Description: codes (s) Returns values so that the construction for p, c in utf8.codes(s) do body end will iterate over all characters in string s, with p being the position (in bytes) and c the code point of each character. It raises an error if it meets any invalid byte sequence.

36.4 Class func - codepoint

Description: codepoint (s [, i [, j]]) Returns the codepoints (as integers) from all characters in s that start between byte position i and j (both included). The default for i is 1 and for j is i. It raises an error if it meets any invalid byte sequence.

36.5 Class func - len

Description: len (s [, i [, j]]) Returns the number of UTF-8 characters in string s that start between positions i and j (both inclusive). The default for i is 1 and for j is -1. If it finds any invalid byte sequence, returns a false value plus the position of the first invalid byte.

36.6 Class func - offset

Description: offset (s, n [, i]) Returns the position (in bytes) where the encoding of the n-th character of s (counting from position i) starts. A negative n gets characters before position i. The default for i is 1 when n is non-negative and #s + 1 otherwise, so that utf8.offset(s, -n) gets the offset of the n-th character from the end of the string. If the specified character is neither in the subject nor right after its end, the function returns nil. As a special case, when n is 0 the function returns the start of the encoding of the character that contains the i-th byte of s. This function assumes that s is a valid UTF-8 string.

37. Lua table functions

37.1 Class func - concat

Description: concat (table, [sep, [i, [j]]]) Given a list where all elements are strings or numbers, returns the string list[i]..sep..list[i+1] ··· sep..list[j]. The default value for sep is the empty string, the default for i is 1, and the default for j is #list. If i is greater than j, returns the empty string.

37.2 Class func - insert

Description: insert (table, [pos], value) Inserts element value at position pos in list, shifting up the elements list[pos], list[pos+1], ⋯, list[#list]. The default value for pos is #list+1, so that a call table.insert(t,x) inserts x at the end of list t.

37.3 Class func - move

Description: move (a1, f, e, t, [a2]) Moves elements from table a1 to table a2, performing the equivalent to the following multiple assignment: a2[t],⋯ = a1[f],⋯,a1[e]. The default for a2 is a1. The destination range can overlap with the source range. The number of elements to be moved must fit in a Lua integer. Returns the destination table a2.

37.4 Class func - pack

Description: pack (...) Returns a new table with all arguments stored into keys 1, 2, etc. and with a field "n" with the total number of arguments. Note that the resulting table may not be a sequence.

37.5 Class func - remove

Description: remove (table, [pos]) Removes from list the element at position pos, returning the value of the removed element. When pos is an integer between 1 and #list, it shifts down the elements list[pos+1], list[pos+2], ⋯, list[#list] and erases element list[#list]; The index pos can also be 0 when #list is 0, or #list + 1; in those cases, the function erases the element list[pos]. The default value for pos is #list, so that a call table.remove(l) removes the last element of list l.

37.6 Class func - sort

Description: sort (table, [comp]) Sorts list elements in a given order, in-place, from list[1] to list[#list]. If comp is given, then it must be a function that receives two list elements and returns true when the first element must come before the second in the final order (so that, after the sort, i < j implies not comp(list[j],list[i])). If comp is not given, then the standard Lua operator < is used instead. Note that the comp function must define a strict partial order over the elements in the list; that is, it must be asymmetric and transitive. Otherwise, no valid sort may be possible. The sort algorithm is not stable: elements considered equal by the given order may have their relative positions changed by the sort.

37.7 Class func - unpack

Description: unpack (table, [i, [j]]) Returns the elements from the given list. This function is equivalent to return table[i], table[i+1], ..., table[j] By default, i is 1 and j is #list.

38. Lua keywords

38.1 Keyword - and

Description: No description available.

38.2 Keyword - break

Description: No description available.

38.3 Keyword - do

Description: No description available.

38.4 Keyword - else

Description: No description available.

38.5 Keyword - elseif

Description: No description available.

38.6 Keyword - end

Description: No description available.

38.7 Keyword - false

Description: No description available.

38.8 Keyword - for

Description: No description available.

38.9 Keyword - function

Description: No description available.

38.10 Keyword - goto

Description: No description available.

38.11 Keyword - if

Description: No description available.

38.12 Keyword - in

Description: No description available.

38.13 Keyword - local

Description: No description available.

38.14 Keyword - nil

Description: No description available.

38.15 Keyword - not

Description: No description available.

38.16 Keyword - or

Description: No description available.

38.17 Keyword - repeat

Description: No description available.

38.18 Keyword - return

Description: No description available.

38.19 Keyword - then

Description: No description available.

38.20 Keyword - true

Description: No description available.

38.21 Keyword - until

Description: No description available.

38.22 Keyword - while

Description: No description available.