Changelist

v1.02 – xxxx/xx/xx
- ✔ Added DEV_reboot function
- ✔ Added CFG_backup2 function with type parameter: user, admin and gallery
- ✔ Bug fix CFG_backup was calling to Device.Backup instead Settings.Backup
- ✔ Typo, Calendar general_enabled changed to bEnabled
- ✔ Typo, Scripts enabled change to bEnabled

v1.01 – 2018/03/14
- ✔ Added DEV_mac function
- ✔ Added DEV_regkey function

v1.00 – 2018/02/08
- ✔ **First version of the document also named “Preliminary version”**

ePLAYER1 LUA SDK

ePLAYER1 is a Lua extension library that acts as interface between LUA and ePLAYER1 firmware using the well known ePLAYER1 JSON protocol . Basically is composed by different kind of objects:

- PLAYER – access player functions
- CFG – access configuration. With this object you can access all the ePLAYER1 configuration.
- PRESET – PRESET settings. There are 20 presets that act as memories where you can store all the ePLAYER1 player settings like url, play mode, repeat mode, fade mode, etc.
- EVENT – Event configuration lets ePLAYER1 execute automated functions. There are 3 possibilities GPI1, GPI2 and SILENCE settings
- CAL – Calendar configuration lets ePLAYER1 execute automated functions on date event. There are up to 24 different calendar.
- SAF – Store and Forward settings
- GDRIVE - lets ePLAYER1 download Google Drive content to local storage devices (USB/MMC). Daily execution on selected time
- CMS – CMS settings
- SCRIPT – Manage ePLAYER1 scripts. Is possbile to execute scripts based on different kind of triggers like BOOT, EVENT, LOAD PRESET, CALENDAR, NETWORK, etc.
- LOG – Register functions that allows the user to write to the ePLAYER1 internal REGISTER
- DEV – Device status and other setups (like install new firmwares)
- PANEL – Panel lock mode settings and status
- FINDER – use FINDER settings to discover your ePLAYER1.
- LCD – Allow access to ePLAYER1 LCD display

How to use ePLAYER1 module?

In order to use ePLAYER1 module you must include it in your source code using require statement. EPLAYER1 automates this function for you, but you can also include ePLAYER1 module manually and create your ePLAYER1 object as you need using the next two lines of code:

```
require "ePLAYER1"
mydev = ePLAYER1.new()
```

In this case you can program actions directly to your ePLAYER1 device, but also it is possible to access ePLAYER1 directly from your PC (in this case it is recommended to use some LUA development tool like Eclipe/LUA)
If you wan to use ePLAYER1 directly from your PC you can call "new" function with IP and PORT parameters as shown next:

```
require "ePLAYER1"
mydev = ePLAYER1.new("10.120.120.4", 2003)
```

Where 10.120.120.4 is the ePLAYER1 IP address, that you can check by pressing ENTER button for 10 seconds and going to WIFI or ETHERNET menu. 2003 is the JSON port.

One time ePLAYER1 object is created is possble to access all functions in it.

Let see what you can do.

Conventions

n indicates number
b indicates boolean
txt indicates text

PLAYER FUNCTIONS

Access player functions like repeat, fade, etc.

Functions

Function:
```
PLAYER_stereo(bStereo)
```
Description:
Set the player output mode to stereo or mono
Parameters:
bStereo – *boolean*, set to true for stereo output, otherwise mono output is selected

Function:
```
PLAYER_fade(nFade)
```
Description:
Set the player fade mode
Arguments:
nFade – *integer* 0-OFF, 1-XFADE, 2-FADE

Function:
```
PLAYER_mode(nPlayMode)
```
Description:
Set the player playlist sequence mode
Arguments:
nPlayMode – *integer* 0-SEQUENTIAL, 1-RANDOM

Function:
```
PLAYER_repeat(nRepeat)
```
Description:
Set the player repeat mode
Arguments:
nRepeat – *integer* 0-PLAY ALL, 1-PLAY ONE, 2-REPEAT ALL, 3-REPEAT ONE

Function:
```
PLAYER_getStats()
```
Description:
Get the player statistics and status
Return:
String – xml string containing all the player status

In the next example you could see how to obtain the player stats and how to parse it using cjson library.

```
require "ePLAYER1"
```

```
ep=ePLAYER1.new()
json_text=ep.PLAYER_getStats()
print(json_text)

cjson=require "cjson"
value=cjson.decode(json_text)
print("SourceList[3]=" .. value.SourceList[3])
```

output

```
{"title":"The White Stripes - Seven Nation Army","counter":"52:52","txtSource":"NET","status":1,"SourceList":
["","MMC","USB","DLNA","AIRPLAY","rock
alternativo"],"source":5,"preset":1,"volume":100,"txtVolume":"0dB","stereo":1,"repeat":2,"playmode":0,"fade":1,"
bootpreset1":0,"sp":1,"bitrate":"192","duration":"--:--","freq":"44.1","playlist_index":" 0001 /
0001","playlist_url":"mmc://radiobob-alternativerock-mp3-hq?
sABC=5n6s2sr8%230%23no8617362n29q2o435p17n54928n16s5%23gharva&amsparams=playerid:tunein;skey:15
17236200","priority":""}
SourceList[3]=USB
```

Function:

```
PLAYER_queue(urlNextElem)
```

Description:

Add next playlist item. Using this function you can compose your own playlist. In order to do a continous play it is necessary to queue next item prior to the end of the current one.

Arguments:

urlNextElem – url of the item to add

Function:

```
PLAYER_priority(urlElemePriority)
```

Description:

Use this function to play priority sounds over the normal playlist. It apply

Return:

String – url of the priority element

Function:

```
PLAYER_play()
```

Description:

If the player is paused or stopped use this function to start current loaded item reproduction, otherwise the player is paused.

Function:

```
PLAYER_stop()
```

Description:

Use this function to stop the current player reproduction.

Function:

```
PLAYER_next()
```

Description:

Use this function to advance to the next item. It retains the play state after advance is done.

Function:

```
PLAYER_previous()
```

Description:

Use this function to move back to the previous item. It retains the play state.

Function:

```
PLAYER_incvol = function()
```

Description:

Use this function to increase the player volume 3dB.

Function:

```
PLAYER_decvol = function()
```

Description:

Use this function to decrease the player volume 3dB.

Function:

```
PLAYER_setvol = function(perc)
```

Description:

Use this function to set the current player volume

Arguments:

nVolume – *integer* 0 to 100

Function:

```
PLAYER_setvol = function(perc)
```

Description:

Call this function to set the mute player parameter.

Arguments:

bMute – *boolean* set to true to mute the player, otherwise unmute the player.

Function:

```
PLAYER_open = function( options )
```

Description:

Call this function to open/load a preset or source or url

Arguments:

url (optional) –- String of the url you want to open e.g. mmc://my-music/ http:://my.server/music-mp3
preset (optional) – integer from 1 to 20. Indicates the preset you want to open
source (optional) – integer from 1 to 24 representing MMC, USB, DLNA, AIRPLAY and presets from PRESET1 to PRESET 20 (preset must be configured properly)

How to load “PRESET 12”?

```
require "ePLAYER1"
ep = ePLAYER1.new()
ep.PLAYER_open({preset=12})
```

How to load url “http:://my.server/music-mp3”?

```
require "ePLAYER1"
ep = ePLAYER1.new()
ep.PLAYER_open({url=”http://my.server/music-mp3”})
```

How to load source “AIRPLAY”?

```
require "ePLAYER1"
ep = ePLAYER1.new()
ep.PLAYER_open({source=4})
```

*note the { } around the parameter which indicates optional arguments

CONFIG (CFG) FUNCTIONS

Set and get ePLAYER1 parameters: PRESET, CMS, SAF, CAL, EVENT, GDRIVE, SCRIPT and LOG. Please refer to the corresponding section to check how to set/get each parameter.

Functions

Function:

```
CFG_reset()
```

Description:

Restore ePLAYER1 configuration to its initial state (factory defaults). Please note that using this function you will lost all your changes.

Function:

```
CFG_restore(urlRestore)
```

Description:

Restore ePLAYER1 settings from the urlRestore file e.g. mmc://good-settings.config http:://my.server/ePLAYER1/mycompany.settings

Arguments:

urlRestore – String indicating the url where settings you want to restore

Note:

If you wan to apply all settings you must call DEV_reboot or call every XXX_reload function to apply changes one per one.

Function:

```
CFG_backup(urlBackup, bUser)
```

Description:

Backups ePLAYER1 settings to urlBackup.

Arguments:

urlBackup – String indicating the url where settings are stored e.g. mmc://good-settings.config http:://my.server/ePLAYER1/mycompany.settings

bUser – Boolean set to true for user settings only, otherwise all the settings are backuped to the target destination

Function:

```
CFG_backup2(urlBackup, type)
```

Description:

Backups ePLAYER1 settings to urlBackup.

Arguments:

urlBackup – String indicating the url where settings are stored e.g. mmc://good-settings.config http:://my.server/ePLAYER1/mycompany.settings

type – String available possibilities are: user, admin and gallery. Select gallery for Player, Events, CMS, Scripts, Player Profile and Network settings. Select user for Player, Presets, Events, Calendar, Cloud Disk, CMS and Scripts. Select Admin for all the available settings.

Function:

```
CFG_get = function(interface, section, variable)
```

Description:

Return the value of the selected variable “interface.section.variable”

Arguments:

interface – must be a valid interface: preset, cms, saf, calendar(01..24), event (gpi1, gpi2, silence), gdrive, script and log.

section – variable's section. Most of the cases is “settings”

variable – variable's name.

Return:
String with the variable value.

Function:
```
CFG_set = function(interface, section, variable, value)
```
Description:
Set the value of the selected variable “interface.section.variable” to “value”
Arguments:
interface – must be a valid interface: preset, cms, saf, calendar(01..24), event (gpi1, gpi2, silence), gdrive, script and log.
section – variable's section. Most of the cases is “settings”
variable – variable's name.
value – variable's value.

Function:
```
CFG_commit = function(interface)
```
Description:
Commit interface changes. After changes all variables in one interface it's necessary to dump changes to ePLAYER1 internal memory. Do it one time for each modified interface.
Arguments:
interface – must be a valid interface: network, wireless, preset, cms, saf, calendar(01..24), event (gpi1, gpi2, silence), gdrive, script and log.

PRESET FUNCTIONS

Set and get ePLAYER1 PRESET parameters. There are 20 presets available. Each preset can configure: name, events enabled, playlist, media alias, play status, volume, mute, play mode, repeat mode, fade mode, stereo/mono.

Functions

Function:
```
PRESET_reload(index)
```
Description:
Reload preset configuration. Call it after modify the preset configuration using CFG_set and CFG_commit
Arguments:
index – number indicating the preset number (from 1 to 20)

Preset variables

presetNN.settings.bname	Preset name
presetNN.settings.eventList	List of enabled events separated by % (GPI1%GPI2%SILENCE)
presetNN.settings.bPlaylist	1 to overwrite playlist
presetNN.settings.loadPlaylist	url of the playlist. Must be enabled using presetNN.settings.bPlaylist
presetNN.settings.mediaAlias	Alias of the enabled playlist. It appears as source
presetNN.settings.bStatus	1 to overwrite status
presetNN.settings.status	Preset status must be PLAY or STOP
presetNN.settings.bVolume	1 to overwrite preset volume
presetNN.settings.volume	Preset volume in % (from 0 to 100)

presetNN.settings.mute	1 indicates mute, 0 to unmute
presetNN.settings.bPlayMode	1 to overwrite play mode
presetNN.settings.playMode	RANDOM or SEQUENTIAL
presetNN.settings.bRepeatMode	1 to overwrite repeat mode
presetNN.settings.repeatMode	PLAY ALL, PLAY ONE, REPEAT ALL or REPEAT ONE
presetNN.settings.bFadeMode	1 to overwrite fade mode
presetNN.settings.fadeMode	OFF, XFADE or FADE
presetNN.settings.bStereo	1 to overwrite stero/mono mode
presetNN.settings.stereo	MONO or STEREO

Note: all variables are optional.
NN indicates a number 01 to 20

Preset examples

```
require "ePLAYER1"
ep = ePLAYER1.new()

ep.CFG_set("preset03", "settings", "bname", "My first preset")
ep.CFG_set("preset03", "settings", "bPlaylist", 1)
ep.CFG_set("preset03", "settings", "mediaAlias", "Alias of my first preset")
ep.CFG_set("preset03", "settings", "bVolume", 1)
ep.CFG_set("preset03", "settings", "volume", 100)
ep.CFG_set("preset03", "settings", "loadPlaylist", "mmc://")
ep.CFG_set("preset03", "settings", "eventList", "GPI1%SILENCE")
ep.CFG_commit("preset03")

ep.PRESET_reload(3)
```

At this moment you can load the configured preset with this instruction:

```
ep.PLAYER_open({preset=3})
```

EVENT FUNCTIONS

Event functions lets ePLAYER1 execute automated functions. There are 3 possibilities GPI1, GPI2 and SILENCE settings.

Functions

Function:

```
EVENT_reload(txtEvent)
```

Description:

Reload event configuration. Call it after modify the preset configuration using CFG_set and CFG_commit

Arguments:

txtEvent – String indicating the event to reload (GPI1, GPI2 or SILENCE)

Event variables

When programmaing GPI1 or GPI2 this variables are available

gpiN.settings.source_polarity	Event polarity DIRECT or REVERSE
gpiN.settings.target_type	INTERNAL, PRESET RECALL, TRANSPORT CONTROL, LOAD & PLAY SOURCE or PRIORITY SOURCE
gpiN.settings.target_preset	If target_type=PRESET RECALL put here the preset number 1 to 20
gpiN.settings.target_transport	If target_type=TRANSPORT CONTROL enter here one of this options: STOP, PLAY, PREV/RW, NEXT/FW
gpiN.settings.target_loadsource	if target_type=LOAD & PLAY SOURCE or target=PRIORITY SOURCE put here the url you want to load and play
gpiN.settings.target_priomode	If target_type=PRIORITY SOURCE enter here the priority mode. Choose between HOLD or PULSE
gpiN.settings.target_prio_retrigger	If target_priomode=PULSE set to 1 to active retrigger option. 0 to disable it
gpiN.settings.target_prio_pulse_time	If target_prio_retrigger=1 put here the playback duration

Note: N should be 1 or 2 (for GPI1 or GPI2)

While programming SILENCE this variables ara available

silence.settings.source_detecttime	Put here the detection time in seconds
silence.settings.target_type	INTERNAL, PRESET RECALL, LOAD & PLAY SOURCE
silence.settings.target_preset	If target_type=PRESET RECALL put here the preset number 1 to 20
silence.settings.target_loadsource	if target_type=LOAD & PLAY SOURCE or target=PRIORITY SOURCE put here the url you want to load and play

CALENDAR (CAL) FUNCTIONS

Calendar functions lets ePLAYER1 execute automated functions on a date/time. There are 24 different calendar events available.

Functions

Function:

```
CAL_reload(nIndex)
```

Description:

Reload calendar configuration. Call it after modify the calendar configuration using CFG_set and CFG_commit

Arguments:

nIndex – number indicating the calendar to reload (1 to 24)

Calendar variables

calendarNN.settings.general_description	Calendar name or description
calendar01.settings.bEnabled	Set to 1 to enable this calendar
calendarNN.settings.source_start_date	Start date (YYYY/MM/DD). Could be blank
calendarNN.settings.source_start_time	Start time (HH:MM). Could be blank.
calendar.settings.source_end_enable	Could be FOREVER or CUSTOM END DATE
calendarNN.settings.source_end_date	If CUSTOM END DATE is selected put here the calendar end date (YYYY/MM/DD)
calendarNN.settings.source_end_time	If CUSTOM END DATE is selected put here the calendar end

	time (HH:MM)
calendarNN.settings.source_week_nonactive	Week mask where calendar **is not** active. Should be a string of days. Each day is represented by his 2 first letters (MoTuWeThFrSaSu)
calendarNN.settings.source_repeat_enable	1 indicates that the calendar must repeat the selected interval time up to the number indicated in times variable. 0 no repetition
calendarNN.settings.source_repeat_interval	If repeat is enabled this parameter indicates the repetition time
calendarNN.settings.source_repeat_times	If repeat is enalbed this parameter indicates the repetition's number
calendarNN.settings.target_type	INTERNAL, PRESET RECALL, TRANSPORT CONTROL, LOAD & PLAY SOURCE or PRIORITY SOURCE
calendarNN.settings.target_preset	If target_type=PRESET RECALL put here the preset number 1 to 20
calendarNN.settings.target_transport	If target_type=TRANSPORT CONTROL enter here one of this options: STOP, PLAY, PREV/RW, NEXT/FW
calendarNN.settings.target_loadsource	if target_type=LOAD & PLAY SOURCE or target=PRIORITY SOURCE put here the url you want to load and play

Note: NN indicates a number 01 to 24

STORE AND FORWARD (SAF) FUNCTIONS

Store and Forward module lets ePLAYER1 download remote media content from a rsync/ssh server to local storage devices (sme as PRESET1), daily, and recall PRESET1 to automatically play it.

Functions

Function:

```
SAF_reload()
```

Description:

Reload Store and Forward configuration. Call it after modify the Store and Forward configuration using CFG_set and CFG_commit

Store and Forward variables

saf.settings.bEnabled	Set to 1 to enable daily Store and Forware synchronization
saf.settings.time_param	Enter here the synchronization hour (HH:MM)
saf.settings.host	Put here the rsync/ssh server address
saf.settings.port	Enter here the rync/ssh port (default 22)
saf.settings.source_path	Enter here the server folder where the contents are stored
saf.settings.source_user	SSH/rsync username parameter
saf.settings.source_key	Enter here your private SSH/rsync key
saf.settings.source_timeout	Enter here the SSH/rsync operation timeout (in seconds)

GOOGLE DRIVE (GDRIVE) FUNCTIONS

GoogleDrive module lets ePLAYER1 download remote media content from a Google Drive account to local storage devices (USB/MMC), daily, and recall PRESET1 to automatically play it.

Functions

Function:
GDRIVE_reload()

Description:
Reload Google Drive configuration. Call it after modify the Store and Forward configuration using CFG_set and CFG_commit

Function:
GDRIVE_authenticate()

Description:
Run the authentication process. Is mandatory to run the authenticate process one time after change the key with CFG_set

Function:
GDRIVE_synchronize()

Description:
Run the synchronization process. It's mandatory to run authenticate process before to call GDRIVE_synchronize (only fist time)

Store and Forward variables

gdrive.settings.bEnabled	Set to 1 to enable daily Google Drive synchronization
gdrive.settings.time_param	Enter here the synchronization hour (HH:MM)
gdrive.settings.source_path	Put here the rsync/ssh server address
gdrive.settings.source_token	Enter here the generated token. Please use this url to generate a token for your Google account
gdrive.settings.target_path	Enter here the server folder where Google Drive contents are stored

CONTENT MANAGEMENT SYSTEM (CMS) FUNCTIONS

CMS lets you automate ePLAYER1 reproduction using a CMS web portal. If you are an ECLER CMS partner you should enter your personal Key in the Key field bellow. Otherwise leave it blank and configure Protocol, Host and Port fields manually to use CMS service with the generic ePLAYER1 CMS API

Functions

Function:
CMS_reload()

Description:
Reload CMS configuration. Call it after modify the Content Management System configuration using CFG_set and CFG_commit

CMS variables

cms.settings.cms_partner_enabled	Set to 1 to enable partner key
cms.settings.cms_key	If cms_partner_enabled is enabled put here your partner key (please contact you ECLER sales rep)
cms.settings.cms_http	If cms_partner_enabled is disabled put here your server protocol (HTTPS or HTTP)
cms.settings.cms_host	Enter here your CMS host address
cms.settings.cms_port	Enter here your CMS host port. If using default port leave it blank

SCRIPTS FUNCTIONS

CMS lets you automate ePLAYER1 reproduction using a CMS web portal. If you are an ECLER CMS partner you should enter your personal Key in the Key field bellow. Otherwise leave it blank and configure Protocol, Host and Port fields manually to use CMS service with the generic ePLAYER1 CMS API

Functions

Function:

```
SCRIPT_reload(nIndex)
```

Description:

Reload selected Script configuration. Call it after modify the Script configuration using CFG_set and CFG_commit

Arguments:

nIndex – number indicating the script index to reload. Must be a number between 1 and 20

Function:

```
SCRIPT_run(nIndex)
```

Description:

Executes the sellected script.

Arguments:

nIndex – number indicating the script index to reload. Must be a number between 1 and 20

Function:

```
SCRIPT_kill(nIndex)
```

Description:

Terminates the sellected script.

Arguments:

nIndex – number indicating the script index to reload. Must be a number between 1 and 20

Function:

```
SCRIPT_status(nIndex)
```

Description:

Retrieves the sellected script status

Arguments:

nIndex – number indicating the script index. Must be a number between 1 and 20

Return:

A XML string with script status {"status":"Idle"} or {"status":"Running"} or {"status":"Success"} or {"status":"Failed"}

Script variables

scriptNN.settings.name	Script name or description

scriptNN.settings.bEnabled	Set to 1 to enable this event trigger
scriptNN.settings.trigger	Script trigger type. Could be one of the next values: ON BOOT, ON EVENT, ON PRESET, ON CALENDAR, ON CLOUD DISK SYNC, ON LAN, ON WAN, ON MMC or ON USB
scriptNN.settings.trigger_event	If trigger is ON EVENT enter here the event that triggers the script. You could put here one of the next values: GPI1, GPI2 or SILENCE
scriptNN.settings.trigger_preset_index	If trigger is ON PRESET enter here the preset number that triggers the script
scriptNN.settings.trigger_calendar	If trigger is ON CALENDAR enter here the calendar number that triggers the script
scriptNN.settings.trigger_cloud	If trigger is ON CLOUD DISK SYNC you could select here RSYNC or GOOGLE DRIVE
scriptNN.settings.trigger_network	If trigger is ON LAN or ON WAN enter here the action that triggers the script. Could be AVAILABLE or UNVAILABLE
scriptNN.settings.trigger_mmc	if trigger is ON MMC enter here the action that triggers the script. Could be PLUG or UNPLUG
scriptNN.settings.trigger_usb	if trigger is ON USB enter here the action that triggers the script. Could be PLUG or UNPLUG

Note: NN indicates a number 01 to 20

REGISTER (LOG) FUNCTIONS

Register functions that allows the user to write to the ePLAYER1 internal REGISTER

Functions

Function:

```
LOG_trace(txtLog)
```

Description:

Add to ePLAYER1 LOG register a trace line

Arguments:

txtLog – String containing the text you want to add. The source of the register line will be equal to “ScriptNN” where NN represents the script calling LOG_trace

Function:

```
LOG_warning(txtLog)
```

Description:

Add to ePLAYER1 LOG register a warning line

Arguments:

txtLog – String containing the text you want to add. The source of the register line will be equal to “ScriptNN” where NN represents the script calling LOG_warning

Function:

```
LOG_error(txtLog)
```

Description:

Add to ePLAYER1 LOG register a error line

Arguments:

txtLog – String containing the text you want to add. The source of the register line will be equal to “ScriptNN” where NN represents the script calling LOG_error

DEVICE (DEV) FUNCTIONS

Device functions allows the user to setup firmware and general ePLAYER1 configurations.

Functions

Function:
```
DEV_reboot()
```
Description:
Reboot the ePLAYER1 inmediatly.

Function:
```
txtVersion = DEV_version()
```
Description:
Get the ePLAYER1 firmware version
Return:
txtVersion – string containing ePLAYER1 firmware version formatted

Function:
```
DEV_update(urlFirmware)
```
Description:
Installs a new ePLAYER1 firmware version. After installation device is rebooted automatically
Arguments:
urlFirmware – Url containing the path where ePLAYER1 firmware resides. Must be a local storage device or http/https url.

Function:
```
total, used, percent = DEV_get(devUrl)
```
Description:
Get the ePLAYER1 firmware version
Arguments:
devUrl – url of local storage device. Must be mmc:// or usb://
Return:
total – number representing total number of bytes of external storage device
used – number of bytes representing the used size
percent – number parameter representing the used percentadge

Example:
```
require "ePLAYER1"
ep=ePLAYER1.new()


print("usb", ep.DEV_get("usb://"))


total,user,percent=ep.DEV_get("mmc://")
print("mmc", total, user, percent)


  usb      7823420        4223365
mmc      15629312        68912   0
```

Function:
```
DEV_boot(nBoot)
```
Description:

Set device boot mode to nBoot

Arguments:

nBoot – number indicating load PRESET1 (bBoot=2) or keep settings (nBoot=1)

Function:

```
bEncrypt = DEV_isEncrypted(devUrl)
```

Description:

Get local storage encrypted flag

Arguments:

devUrl – string must be mmc:// or usb://

Return:

bEncrypt – boolean, true, if local storage is encrypted

Function:

```
bEncrypt = DEV_isEncrypted(devUrl)
```

Description:

Get local storage encrypted flag

Arguments:

devUrl – string must be mmc:// or usb://

Return:

bEncrypt – boolean, true, if local storage is encrypted

Function:

```
bEncrypt = DEV_mac()
```

Description:

Get device MAC address

Return:

jsonMAC – json string representing MAC address value

```
require "ePLAYER1"
device=ePLAYER1.new()
print(device.DEV_mac())
```

```
{"mac":"A8 40 41 16 47 D6"}
```

Function:

```
bEncrypt = DEV_regkey()
```

Description:

Get device Registration Key code

Return:

jsonMAC – json string representing Registration Key code

```
require "ePLAYER1"
device=ePLAYER1.new()
print(device.DEV_regkey())
```

```
{"regkey":"A269FCEAB4F1C20B"}
```

PANEL FUNCTIONS

Panel functions allows the user to setup firmware and general ePLAYER1 configurations.

Functions

Function:
```
PANEL_set=function(lockMode, password)
```
Description:
Set the panel lock mode and password
Arguments:
lockMode – string indicating lock mode. Possible options are UNLOCK ALL, UNLOCK USER, LOCK ALL
password – string with the LOCK/UNLOCK password. Max length is 8. Valid characters are: 0-9 A-Z . + -

Function:
```
lockMode, password = PANEL_get()
```
Description:
Set the panel lock mode and password
Return:
lockMode – string representing lock mode. Possible values are UNLOCK ALL, UNLOCK USER, LOCK ALL
password – string with the LOCK/UNLOCK password

FINDER FUNCTIONS

Finder functions allows the user to run finder operation on ePLAYER1.

Functions

Function:
```
FINDER_set(bLigth)
```
Description:
Start or Stop finder operation. If finder is active, ePLAYER1 display blinks.
Arguments:
bLigth – boolean, set to true to start finder operation (display blink). Set to 0 to stop finder operation (normal state)

LCD FUNCTIONS

LCD functions allows the user to show messages on ePLAYER1 LCD display. The texts are limited to LCD physical dimensions: 2 lines per 16 columns

Functions

Function:
```
LCD_print(txtLine1, txtLine2, center, timeout)
```
Description:
This functions shows a message on the ePLAYER1 LCD display
Arguments:
txtLine1 – String corresponding to the text on the first LCD line
txtLine2 – String corresponding to the text on the second LCD line
center – Booelan, set to true to center the text on the LCD display
timeout – Number of seconds that message is showed in ePLAYER1 LCD display

Function:

```
LCD_print2({txtLine1, txtLine2, bCenter, nTimeout})
```

Description:

This functions shows a message on the ePLAYER1 LCD display. Parameters are optional.

Arguments:

txtLine1 (optional) – String corresponding to the text on the first LCD line
txtLine2 (optional)– String corresponding to the text on the second LCD line
center (optional) – Booelan, set to true to center the text on the LCD display
nTimeout (optional) – Number of seconds that message is showed in ePLAYER1 LCD display