xxter LUA-scripts handleiding

xxter scripts introductie

Met xxter scripts kunt u eenvoudig zelf kleine programma's maken binnen xxter. Deze scripts zijn erg flexibel en kunnen vele mogelijkheden toevoegen aan een bestaande domotica-installatie. Zo kunt u logische functies maken, acties vertraagd laten uitvoeren, scènes uitbreiden met verlopende RGB-verlichting en veel meer. Scripts kunnen worden geactiveerd door een scène, een planner, een actie (trigger) of een ander script.

xxter ondersteunt zowel een "native" scripttaal als LUA-scripts. Deze handleiding geeft de uitleg over de LUA-scripts. Meer informatie over de xxter native scripts vindt u op de documentatie pagina van onze website (https://www.xxter.com/documentation/). Er zijn ook verschillende voorbeelden van scripts beschikbaar op deze pagina en op ons forum (https://forum.xxter.com).

Om meer te weten te komen over LUA in het algemeen, is de volgende handleiding beschikbaar: https://www.lua.org/manual/5.3/

De basis

xxter LUA-scripts zijn geschreven programma's die uit een of meerdere regels bestaan. U kunt commentaar regels aan een script toevoegen door "--" te schrijven. Dit is mogelijk op een aparte regel, of aan het einde van een regel na een commando.

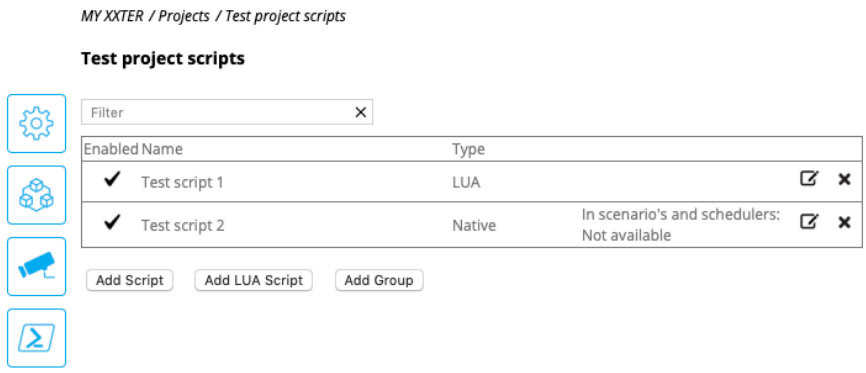
Voorbeeld van een xxter LUA-script:

```
1  -- this is an example test script.
2  xxter.userlog("starting test script")
3  -- Setting dimmer to 50%
4  xxter.setcomponent(16, 50)
5  -- Waiting for 5 seconds
6  xxter.sleep(5000)
7  -- Setting switch on
8  xxter.setcomponent(56, 1)
9
```

Scripts kunnen worden gestart en gestopt op verschillende manieren, bijvoorbeeld door een scene of een planner. Scripts kunnen "oneindig" zijn en op herhalende intervallen acties uitvoeren of worden gedefinieerd als een eenmalige reeks commando's die achtereenvolgens worden uitgevoerd, steeds wanneer het script wordt gestart. Wanneer u een "oneindig" script maakt, dat zich continu herhaalt, let er dan op dat u altijd een "sleep" periode inbouwt, om te voorkomen dat het script een te zware belasting op het xxter apparaat heeft.

Scripts beheren

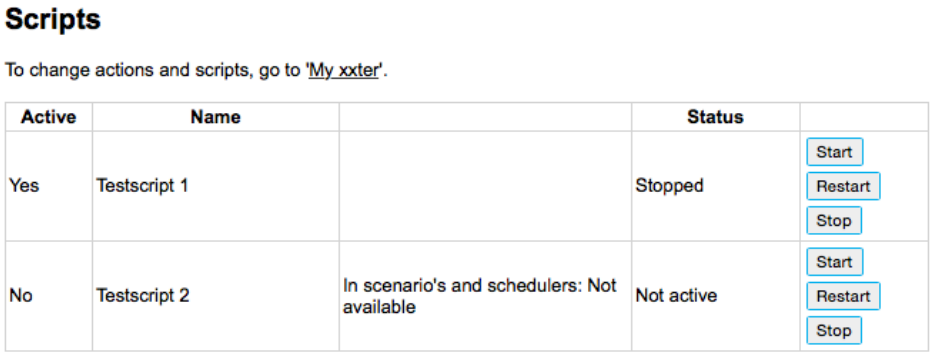
xxter scripts kunnen worden gemaakt en aangepast via de xxter script editor. Deze editor kunt u vinden in de projectconfiguratie van “Mijn xxter”. Selecteer het project waarvoor u scripts wilt beheren en klik in de menubalk op “Scripts”. Alle huidige scripts worden hier weergegeven en kunnen worden bewerkt en verwijderd. Bij het maken van een nieuw script heeft u de keuze om een native xxter script of een LUA-script te maken.



Belangrijk: Als u scripts toevoegt of aanpast, moeten deze worden ingeladen op het xxter apparaat voordat ze kunnen worden gebruikt.

Wanneer er een wijziging is uitgevoerd, ziet u rechtsboven het project de tekst “Project is gewijzigd”. Als u hierop klikt en vervolgens op “Verzoek apparaat om configuratie op te halen”, dan zal het xxter apparaat de nieuwe configuratie ophalen. Dit kunt u alleen gebruiken wanneer xxter is ingesteld om buitenshuis te gebruiken (zie Installatiehandleiding hoofdstuk 13). Als alternatief kunt u ook het profiel opnieuw inladen vanuit de app (zie gebruikershandleiding hoofdstuk 11). Uiteraard kunt u ook het profiel opnieuw inladen vanaf het xxter apparaat zelf, door in te loggen op het apparaat en linksboven in het menu op de knop “Configuratie laden” te drukken.

Wanneer u op het xxter apparaat bent ingelogd, kunt u de ingeladen scripts zien, door in het menu de optie “Scripts” te kiezen. Hier kunnen scripts ook handmatig worden gestart, herstart of gestopt voor testdoeleinden.

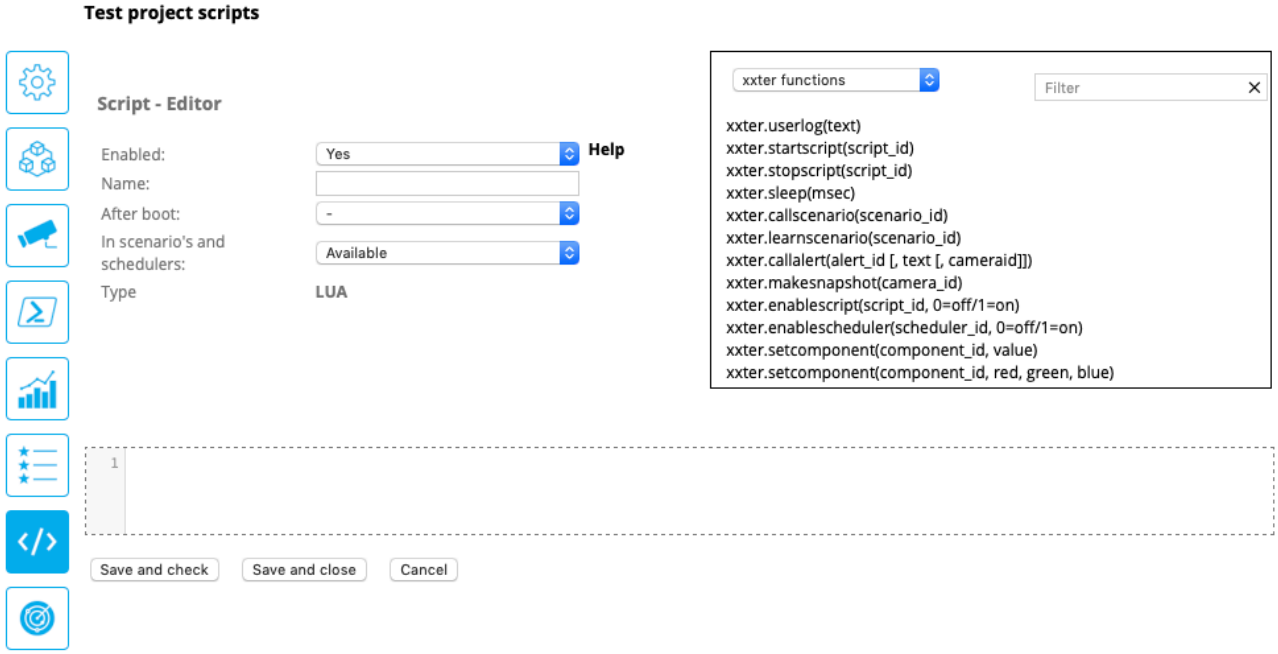


Voor probleemoplossing is de logfunctie van het apparaat erg handig. Op het apparaat kunt u “Scripts” aanzetten onder het kopje “Gebruikerslogboek” op de *Instellingen– basis* pagina. Dit zorgt ervoor dat iedere uitgevoerde regel van ieder script in het logboek wordt weggeschreven. Het logboek kunt u openen via “Open het gebruikerslogboek” op de *Status* pagina.

Scripts editor

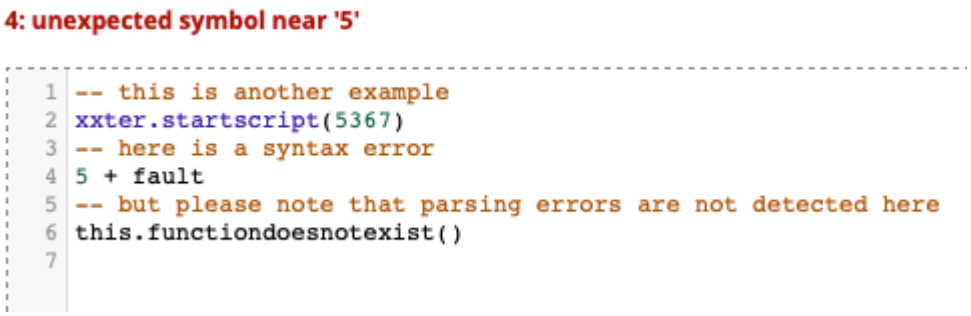
Als u een script toevoegt of bewerkt, kunt u aangeven of deze actief is of niet. Ook kunt u aangeven of het script voor de eindgebruiker beschikbaar is voor het gebruik in scènes of de planner. Daarnaast kunt u aangeven of het scripts automatisch moet worden gestart, wanneer het xxter apparaat (opnieuw) wordt gestart.

Let op: Gedeactiveerde scripts kunnen niet worden uitgevoerd, ook niet als ze worden aangeroepen vanuit een scène, actie of planner. Dit kan handig zijn bij het testen van scripts of bij het verhelpen van problemen. Houd er echter rekening mee dat een script wel kan worden geactiveerd of gedeactiveerd door een ander script.

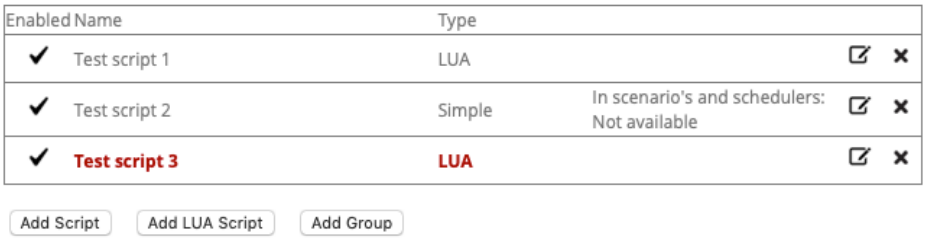


De regelnummers die naast het script worden weergegeven staan er alleen ter informatie en worden in het script niet gebruikt. Commando's kunnen worden ingetypt, maar ook worden ingevoegd via het commandovenster, rechtsboven op het scherm. Let bij het invoegen van commando's op dat de cursor zich op de juiste plek in de script editor bevindt. Bij het invoegen van een commando uit het commandovenster moeten de variabelen nog worden ingesteld met de juiste waarden. De commando's zullen informatie bevatten wat voor variabelen er vereist zijn. De ID-nummers van de componenten kunnen ook uit het commandovenster worden ingevoegd, door bovenaan het type van de waarde te selecteren dat u nodig heeft, bijvoorbeeld "scripts" of "bytes". Door op een component in de lijst te klikken, wordt deze in het script gezet op de plaats van de cursor.

Nadat een of meerdere commando's zijn toegevoegd via de editor, kan worden gecontroleerd of deze juist zijn ingegeven door op de "Opslaan en controleren" knop te drukken. Wanneer er een syntaxfout in het script zit, wordt dit met een melding getoond boven het bewerkvenster. Let op: parse-fouten (bijvoorbeeld de validatie of een functienaam correct is geschreven of het juiste aantal variabelen bevat) worden echter niet opgemerkt door de editor.



Wanneer u een regel heeft gecorrigeerd of toegevoegd kunt u eenvoudig met dezelfde knop het script nogmaals controleren. Met de "Opslaan en sluiten" knop kan de editor worden afgesloten. Scripts die worden opgeslagen met een fout worden rood weergegeven in de lijst.



Opmerking: scripts met een fout kunnen worden opgeslagen, maar kunnen niet worden uitgevoerd.

Wanneer het laatste commando aan het einde van een script is uitgevoerd, stop het script automatisch.

Scripts worden geschreven met behulp van functies en waarden. Daarnaast kunnen scripts worden uitgebreid met variabelen en controlestructuren zoals IF-statements en WHILE-loops.

Om meer te leren over LUA kunt u ook gebruik maken van de algemene LUA-handleiding: https://www.lua.org/manual/5.3/

xxter functies

De volgende xxter LUA functies zijn beschikbaar:

---

xxter.userlog(text)

Beschrijving:
Schrijft tekst naar het gebruikerslogboek, handig voor het debuggen van LUA-scripts. Deze opdracht schrijft altijd naar het log, ook wanneer het loggen van scripts staat uitgeschakeld op de *Instellingen – Basis* pagina, zie pagina 3 van deze handleiding.

Parameters:
text : Een string met tekst die naar het logboek moet worden geschreven.

Returns: -

Voorbeeld:
```
xxter.userlog("Running script part X")
```

---

xxter.startscript(script_id)

Beschrijving:
Start een ander xxter script, wanneer deze niet al draait.

Parameters:
script_id : Het ID van het script (nummer)

Returns: -

Voorbeeld:
```
xxter.startscript(35)
```

---

xxter.stopscript(script_id)

Beschrijving:
Stopt een draaiend xxter script.

Parameters:
script_id : Het ID van het script (nummer)

Returns: -

Voorbeeld:
```
xxter.stopscript(35)
```

---

xxter.enablescript(script_id, status)

Beschrijving:
Activeert of deactiveert een xxter script, zodat deze (niet) kan worden gestart.

Parameters:
script_id : Het ID van het script (nummer)
status : Gewenste script status (enum: 0 = uit, 1 = aan)

Returns: -

Voorbeeld:
```
xxter.enablescript(35,1)
```

---

xxter.sleep(time)

Beschrijving:
Wacht de opgegeven tijd (in milliseconden), voordat het script verder gaat.

Parameters:
time : Het aantal milliseconden dat gewacht moet worden (nummer)

Returns: -

Voorbeeld:
```
xxter.sleep(5000)
```

---

xxter.callscenario(scenario_id)

Beschrijving:
Roept een xxter scene af om de geconfigureerde acties uit te voeren.

Parameters:
scenario_id : Het ID van de scene (nummer)

Returns: -

Voorbeeld:
```
xxter.callscenario(17)
```

---

xxter.learnscenario(scenario_id)

Beschrijving:
Update de geconfigureerde acties van de betreffende xxter scene naar hun huidige waarden

Parameters:
scenario_id : Het ID van de scene (nummer)

Returns: -

Voorbeeld:
```
xxter.learnscenario(17)
```

xxter.callalert(alert_id[, text [,camera_id]])

Beschrijving:
Roept de xxter waarschuwingsservice af, optioneel met opgegeven tekst en camera snapshot. De (optionele) tekst die wordt meegeven wordt geplaatst in de “[x]” positie van de waarschuwing, indien aanwezig.

Parameters:
alert_id : Het ID van de waarschuwing (nummer)
text : *Optioneel,* de tekst die met de waarschuwing wordt gezonden (string)
camera_id : *Optioneel,* het ID van de camera voor de snapshot (nummer)

Returns: -

Voorbeelden:
```
xxter.callalert(65)
xxter.callalert(65, "Scripted alert")
xxter.callalert(65, "Scripted alert", 103)
```

---

xxter.makesnapshot(camera_id)

Beschrijving:
Maakt een snapshot van de opgegeven camera.

Parameters:
camera_id : Het ID van de camera voor het maken van de snapshot (nummer)

Returns: -

Voorbeelden:
```
xxter.makesnapshot(103)
```

---

xxter.enablescheduler(scedule_id, status)

Beschrijving:
Activeert of deactiveert de opgegeven xxter planner, zodat deze (niet) kan starten.

Parameters:
scedule_id : Het ID van de planner (nummer)
status : Gewenste planner status (enum: 0 = uit, 1 = aan)

Returns: -

Voorbeeld:
```
xxter.enablesceduler(18,1)
```

---

xxter.setcomponent(component_id, value [, value_2, value_3])

Beschrijving:
Zet een component op de gegeven waarde.

Parameters:
component_id : Het ID van het component (nummer)
value : Gewenste waarde van het component (type afhankelijk van component).
Wanneer component RGB is, dan is dit de R-waarde (nummer)
value_2 : *Alleen voor RGB,* betreft de G-waarde (nummer)
value_3 : *Alleen voor RGB,* betreft de B-waarde (nummer)

Returns: -

Voorbeelden:
```
xxter.setcomponent(38,75)
xxter.setcomponent(45,75,230,176)
```

---

xxter.getcomponent(component_id)

Beschrijving:
Haalt de huidige waarde op van een component.

Parameters:
component_id : Het ID van het component (nummer)

Returns:
value : Huidige waarde van het component (type afhankelijk van component).
RGB-componenten, geven drie waarden terug

Voorbeeld:
```
var = xxter.getcomponent(38)
varR, varG, varB = xxter.getcomponent(45)
```

---

xxter.readcomponent(component_id)

Beschrijving:
Voert een read commando uit op de KNX-bus voor het component. Wanneer het KNX-component reageert op het verzoek zal xxter de huidige waarde updaten (dit kan even duren). Deze bijgewerkte waarde kan vervolgens door het script worden opgevraagd met *xxter.getcomponent*.

Parameters:
component_id : Het ID van het component (nummer)

Returns: -

Voorbeeld:
```
xxter.readcomponent(38)
```

---

xxter.httpcommand(http_id)

Beschrijving:
Voert het betreffende HTTP-commando uit.

Parameters:
http_id : Het ID van het HTTP-commando (nummer)

Returns: -

Voorbeeld:
```
xxter.httpcommand(12)
```

---

xxter.tcpcommand(tcp_id)

Beschrijving:
Voert het betreffende TCP-commando uit.

Parameters:
tcp_id : Het ID van het TCP-commando (nummer)

Returns: -

Voorbeeld:
```
xxter.tcpcommand(17)
```

---

xxter.ircommand(ir_id)

Beschrijving:
Voert het betreffende infraroodcommando uit.

Parameters:
ir_id : Het ID van het IR-commando (nummer)

Returns: -

Voorbeeld:
```
xxter.ircommand(7)
```

---

xxter.openKNXtunnel(status)

Beschrijving:
Opent of sluit de KNX-tunnel.

Parameters:
status : Commando voor het openen of sluiten van de tunnel (enum; 0=dicht, 1=open)

Returns: -

Voorbeeld:
```
xxter.openKNXtunnel(1)
```

xxter.setsimulation(status)

Beschrijving:
Stelt de aanwezigheidssimulatie in op de gewenste stand.

Parameters:
status : Gewenste stand van de aanwezigheidssimulatie (enum; 0=stop, 1=record, 2=play)

Returns: -

Voorbeeld:
```
xxter.setsimulation(2)
```

---

xxter.getsimulation()

Beschrijving:
Haalt de huidige stand van de aanwezigheidssimulatie op.

Parameters: -

Returns:
Huidige stand van de aanwezigheidssimulatie (enum; 0=stop, 1=record, 2=play)

Voorbeeld:
```
var = xxter.getsimulation()
```

---

xxter.connectKNX(status)

Beschrijving:
Verbinden of verbreken van de KNX-verbinding van het apparaat.

Parameters:
status : Commando voor de KNX-verbinding (enum; 0=disconnect, 1=connect)

Returns: -

Voorbeeld:
```
xxter.connectKNX(1)
```

---

xxter.setpersistent(varname, value)

Beschrijving:
Een variabele aanmaken of updaten dat persistent is (aanwezig is) voor alle scripts, om zo tussen scripts informatie door te geven.

Parameters:
varname : Naam van de persistente variabele (string, alfanumeriek: 0-9a-zA-Z)
value : Waarde voor de persistente variabele (type afhankelijk van de variabele)

Returns: -

Voorbeelden:
```
xxter.setpersistent("maxValue",21.23)
xxter.setpersistent("message","Valve error")
```

---

xxter.getpersistent(varname)

Beschrijving:
Haalt de waarde op van een variabele die persistent is (aanwezig is) voor alle scripts, om zo tussen scripts informatie door te geven.

Parameters:
varname : Naam van de persistente variabele (string, alfanumeriek: 0-9a-zA-Z)

Returns:
Waarde van de persistente variabele (type afhankelijk van de variabele)

Voorbeelden:
```
var = xxter.getpersistent("maxValue")
```

```
xxter.setsonos(device_id, command
                        [, option_setting])
```

Beschrijving:
Uitvoeren van een Sonos commando

Parameters:
device_id : Kan ofwel “global” zijn voor generieke commando’s, of voor een specifiek Sonos device ID (format: RINCON_949F3EC192E401400).
command: Wanneer het device_id “global” is, bestaan de volgende commando’s:

- “creategroup” : vereist aanvullende option_setting met het group_id (nummer)
- “muteall”
- “unmuteall”
- “pauseall”
- “playall”

Wanneer het device_id een specifiek Sonos device is, bestaan de volgende commando’s:

- “groupvolume” : vereist option_setting voor volume (nummer)
- “groupmute”
- “volume” : vereist option_setting voor volume (nummer)
- “mute”
- “unmute”
- “play”
- “pause”
- “seek” : vereist option_setting voor positie (nummer)
- “next”
- “previous”
- “selectplaylist” : vereist option_setting voor de naam van de playlist (string)
- “selectfavorite” : vereist option_setting voor de naam van de favorite (string)
- “selectnextfavorite”
- “selectlinein”
- “selectHDMI”
- “playaudioclip” : vereist twee option_settings, één voor het type audio clip (enum) en één voor het clip nummer (nummer). Audio clip types zijn:
  - 0: Sonos standaard
  - 1: xxter standaard
  - 2: custom

optional_setting : Afhankelijk van het gebruikte commando (zie parameters boven)

Returns: -

Voorbeelden:

```
xxter.setsonos("global", "creategroup", 3)
xxter.setsonos("global", "groupmute")
xxter.setsonos("RINCON_949F3EC192E401400", "selectfavorite", "Radio 4")
xxter.setsonos("RINCON_949F3EC192E401400", "playaudioclip", 2, 4)
```

xxter.getsonos(device_id, parameter)

Beschrijving:
Haalt de huidige status van een Sonos device op

Parameters:
device_id : Sonos device ID (format: RINCON_949F3EC192E401400)
parameter : De parameter die moet worden teruggegeven (een van de volgende opties:
"groupvolume", "volume", "status", "mute", "groupmute", "meta")

Returns:
groupvolume : huidig volume van de Sonos group of het device (nummer)
volume : huidig volume van het Sonos device (nummer)
status : huidige status van het Sonos device (enum: 0:IDLE, 1:BUFFER, 2:PLAY, 3:PAUSED)
(group)mute : huidige (group) mute status (1 als muted, 0 als niet muted)
a, b, c, d, e, f, g : meta data van de spelende titel (meerdere strings)
A = track_name
B = track_artist
C = album_name
D = album_artist
E = show_name
F = container_name
G = stream_info

Voorbeelden:
```
groupvol = xxter.getsonos("RINCON_949F3EC192E401400", "groupvolume")
a, b, c, d, e, f, g = xxter.getsonos("RINCON_949F3EC192E401400", "meta")
```

---

xxter.setupnp(device_id, command[, option_setting])

Beschrijving:
Voert een uPnP-commando uit

Parameters:
device_id : uPnP device ID (format: RINCON_949F3EC192E401400)
command: de volgende commando's kunnen worden uitgevoerd:
"volume" : vereist option_setting voor volume (nummer)
"mute"
"unmute"
"play"
"pause"
"seek" : vereist option_setting voor positie (nummer)
"next"
"previous"

optional_setting : afhankelijk van het gebruikte commando (zie parameters boven)

Returns: -

Voorbeelden:
```
xxter.setupnp("RINCON_949F3EC192E401400", "volume", 25)
xxter.setupnp("RINCON_949F3EC192E401400", "mute")
```

---

xxter.getupnp(device_id)

Beschrijving:
Haalt de huidige status van een uPnP device op

Parameters:
device_id : uPnP device ID (format: RINCON_949F3EC192E401400)
parameter : parameter die moet worden teruggegeven (een van deze: “volume”, “status”, “meta”)

Returns:
volume : huidig volume van het uPnP device (nummer)
status : huidige status van het uPnP device (string)
a, b, c, d, e, f, g : meta data van de spelende titel (meerdere strings)
A = track_name
B = track_artist
C = album_name
D = album_artist
E = show_name
F = container_name
G = stream_info

Voorbeelden:
```
volume = xxter.getupnp("RINCON_949F3EC192E401400", "volume")
a, b, c, d, e, f, g = xxter.getupnp("RINCON_949F3EC192E401400", "meta")
```

---

xxter.timezonediff()

Beschrijving:
Geeft het verschil in seconden tussen de lokale tijdzone en UTC (GMT zonder zomertijd).

Parameters: -

Returns:
Verschil tussen tijdzone en UTC in seconden

Voorbeelden:
```
var = xxter.timezonediff()
```

---

xxter.localtime()

Beschrijving:
Geeft de lokale tijd in seconden. Het gebruik lijkt op os.clock() maar werkt op de lokale tijdzone in plaats van UTC (GMT zonder zomertijd). Deze functie kan bijvoorbeeld worden gebruikt samen met de LUA-functie os.date(format, time) als de time variabele.

Parameters: -

Returns:
Lokale tijd in seconden

Voorbeelden:
```
var = xxter.localtime()
```

Ondersteunde algemene LUA-functies

Naast de bovenstaande xxter functies worden de volgende standaard LUA-functies ondersteund:

```
tonumber (e [, base])
tostring (v)
type (v)
coroutine.create (f)
coroutine.isyieldable ()
coroutine.resume (co [, val1, ···])
coroutine.running ()
coroutine.status (co)
coroutine.wrap (f)
coroutine.yield (···)
string.byte (s [, i [, j]])
string.char (···)
string.dump (function [, strip])
string.find (s, pattern [, init
                        [, plain]])
string.format (formatstring, ···)
string.gmatch (s, pattern)
string.gsub (s, pattern, repl [, n])
string.len (s)
string.lower (s)
string.match (s, pattern [, init])
string.pack (fmt, v1, v2, ···)
string.packsize (fmt)
string.rep (s, n [, sep])
string.reverse (s)
string.sub (s, i [, j])
string.unpack (fmt, s [, pos])
string.upper (s)
utf8.char (···)
utf8.charpattern
utf8.codes (s)
utf8.codepoint (s [, i [, j]])
utf8.len (s [, i [, j]])
utf8.offset (s, n [, i])
os.clock ()
os.date ([format [, time]])
```

```
os.difftime (t2, t1)
os.time ([table])
table.concat (list [, sep [, i [, j]]])
table.insert (list, [pos,] value)
table.move (a1, f, e, t [,a2])
table.pack (···)
table.remove (list [, pos])
table.sort (list [, comp])
table.unpack (list [, i [, j]])
math.abs (x)
math.acos (x)
math.asin (x)
math.atan (y [, x])
math.ceil (x)
math.cos (x)
math.deg (x)
math.exp (x)
math.floor (x)
math.fmod (x, y)
math.huge
math.log (x [, base])
math.max (x, ···)
math.maxnumber
math.min (x, ···)
math.minnumber
math.modf (x)
math.pi
math.rad (x)
math.random ([m [, n]])
math.randomseed (x)
math.sin (x)
math.sqrt (x)
math.tan (x)
math.tonumber (x)
math.type (x)
math.ult (m, n)
```

Documentatie voor deze functies staan in de LUA-handleiding: https://www.lua.org/manual/5.3/