



xxter Lua-scripts handleiding

XXTER SCRIPTS INTRODUCTIE	2
XXTER FUNCTIES	6
ONDERSTEUNDE ALGEMENE LUA-FUNCTIES	19

xxter scripts introductie

Met xxter scripts kan je eenvoudig zelf kleine programma's maken binnen xxter. Deze scripts zijn erg flexibel en kunnen vele mogelijkheden toevoegen aan een bestaande domotica-installatie. Zo kan je 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 vind je 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 één of meerdere regels bestaan. Je 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 je een "oneindig" script maakt, dat zich continu herhaalt, let er dan op dat je altijd een "sleep" periode inbouwt, om te voorkomen dat het script een te zware belasting op de xxter controller geeft.

Scripts beheren

xxter scripts kunnen worden gemaakt en aangepast via de xxter script editor. Deze editor kan je vinden in de projectconfiguratie van “Mijn xxter”. Selecteer het project waarvoor je 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 heb je de keuze om een native xxter script of een Lua-script te maken.

MY XXTER / Projects / Test project scripts

Test project scripts



Enabled	Name	Type		
☒	Test script 1	LUA		
☒	Test script 2	Native	In scenario's and schedulers: Not available	



Belangrijk: Als je scripts toevoegt of aanpast, moeten deze worden ingeladen op de xxter controller voordat ze kunnen worden gebruikt.

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

Wanneer je op de xxter controller bent ingelogd, kan je 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.

Scripts

To change actions and scripts, go to '[My xxter](#)'.

Active	Name		Status	
Yes	Testscript 1		Stopped	Start Restart Stop
No	Testscript 2	In scenario's and schedulers: Not available	Not active	Start Restart Stop

Voor probleemoplossing is de logfunctie van de xxter controller erg handig. Op de xxter controller kan je “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 is te openen via “Open het gebruikerslogboek” op de *Status* pagina.

Scripts editor

Als je een script toevoegt of bewerkt, kan je aangeven of deze actief is of niet. Ook kan je aangeven of het script voor de eindgebruiker beschikbaar is voor het gebruik in scènes of de planner. Daarnaast kan je aangeven of het scripts automatisch moet worden gestart, wanneer de xxter controller (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.

Test project scripts










Script - Editor

Enabled:  **Help**

Name:

After boot: 

In scenario's and schedulers: 

Type: **LUA**


 

```

xxter.userlog(text)
xxter.startscript(script_id)
xxter.stopscrip(script_id)
xxter.sleep(msec)
xxter.callscenario(scenario_id)
xxter.learnscenario(scenario_id)
xxter.callalert(alert_id [, text [, cameraid]])
xxter.makesnapshot(camera_id)
xxter.enablescript(script_id, 0=off/1=on)
xxter.enablescheduler(scheduler_id, 0=off/1=on)
xxter.setcomponent(component_id, value)
xxter.setcomponent(component_id, red, green, blue)

```

1

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 je nodig hebt, bijvoorbeeld "scripts", "integers", "floating points", etc. Door op een component in de lijst te klikken, wordt deze in het script gezet op de plaats van de cursor.


 

1 - Test light one
2 - Test light two

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) kunnen niet worden opgemerkt door de editor.

4: unexpected symbol near '5'

```

1  -- this is another example
2  xxter.startscript(5367)
3  -- here is a syntax error
4  5 + fault
5  -- but please note that parsing errors are not detected here
6  this.functiondoesnotexist()
7

```

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

Enabled Name		Type			
✓	Test script 1	LUA			
✓	Test script 2	Simple	In scenario's and schedulers: Not available		
✓	Test script 3	LUA			

Add Script

Add LUA Script

Add Group

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. Ook is het mogelijk om eigen functies te definiëren met input parameters en return waarden.

Om meer te leren over Lua kan je 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.include(script_id)
```

Beschrijving:

Neemt alle code uit het opgegeven script over in het huidige script. Hiermee kan bijvoorbeeld een set functies eenmalig gedefinieerd worden in één script en in meerdere scripts worden gebruikt.

Parameters:

script_id : Het ID van het script (nummer)

Returns: -

Voorbeeld:

```
xxter.include(17)
```

```
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 meegegeven 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(schedule_id, status)
```

Beschrijving:

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

Parameters:

schedule_id : Het ID van de planner (nummer)

status : Gewenste planner status (enum: 0 = uit, 1 = aan)

Returns: -

Voorbeeld:

```
xxter.enablescheduler(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 [, result])
```

Beschrijving:

Voert het betreffende HTTP-commando uit.

Parameters:

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

result : Optionele parameter om ook het resultaat van de opdracht op te vragen (boolean)

Returns:

result : Indien opgevraagd en beschikbaar, het resultaat dat wordt teruggegeven door het HTTP-commando (string)

Voorbeeld:

```
xxter.httpcommand(12)  
var = xxter.httpcommand(13,true)
```

```
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.getSunAzimuth()
```

Beschrijving:

Haalt de huidige zonnehoek op ten opzichte van het noorden (draaihoek), gebaseerd op de ingestelde locatie van de xxter controller.

Parameters: -

Returns:

Waarde van de draaihoek van de zon.

Voorbeelden:

```
var = xxter.getSunAzimuth()
```

```
xxter.getSunAltitude()
```

Beschrijving:

Haalt de huidige zonnehoek op ten opzichte van de horizon (hoogtehoek), gebaseerd op de ingestelde locatie van de xxter controller.

Parameters: -

Returns:

Waarde van de hoogtehoek van de zon.

Voorbeelden:

```
var = xxter.getSunAltitude()
```

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

Beschrijving:

Uitvoeren van een Sonos of BluOS commando

Parameters:

device_id: Kan ofwel "global" zijn voor generieke commando's, of voor een specifiek 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 audio 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/BluOS standaard
 1: xxter standaard
 2: custom

optional setting : Afhankelijk van het gebruikte commando (zie parameters boven)

Returns: -

Voorbeelden:

```

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

Voorbeeld 1 en 2 zijn generieke commando's, om te groeperen of alle speakers op mute te zetten. Voorbeeld 3 en 4 zijn voor één speaker, om "Radio 4" te spelen of een custom audioclip te spelen.

`xxter.getaudio(device_id, parameter)`

Beschrijving:

Haalt de huidige status van een Sonos/BluOS device op

Parameters:

device_id : Sonos/BluOS 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 group of het device (nummer)

volume : huidig volume van het audio device (nummer)

status : huidige status van het audio 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.getaudio("RINCON_949F3EC192E401400", "groupvolume")
a, b, c, d, e, f, g = xxter.getaudio("RINCON_949F3EC192E401400", "meta")
  
```

Voorbeeld 1 vraagt het groepvolume op, voorbeeld 2 de metadata van het huidige nummer.

```
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")
```

Voorbeeld 1 stelt het volume van een speaker in, voorbeeld 2 zet de speaker op mute.

```
xxter.getupnp(device_id, parameter)
```

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")
```

Voorbeeld 1 vraagt het volume op, voorbeeld 2 de metadata van het huidige nummer.

```
xxter.timezonediff()
```

Beschrijving:

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

Parameters: -

Returns:

Vershil 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()
```

```
xxter.getmeteostatus()
```

Beschrijving:

Geeft het aantal uren sinds de weersinformatie het laatst is opgehaald. De weersinformatie wordt vanzelf bijgewerkt, deze functie is beschikbaar om eventueel de status te kunnen opvragen.

Parameters: -

Returns:

Aantal uren sinds laatste weersupdate

Voorbeelden:

```
var = xxter.getmeteostatus()
```

```
xxter.getmeteoinfo(what [, offset [, unit]])
```

Beschrijving:

Deze functie geeft de gevraagde waarde van de gevraagde eigenschap terug.

Wanneer de data niet beschikbaar is, wordt *false* terug gegeven.

Parameters:

what : Eigenschap die moet worden opgevraagd, zie tabel 1 hieronder (string)

offset : Tijdsverschuiving in uren, hoever in de toekomst de informatie moet zijn (number)

Bijvoorbeeld een "4" geeft de weersinformatie verwacht over 4 uur. Default is 0.

unit : Eenheid van de gevraagde gegevens (enum; "C"=Celsius, "K"=Kelvin, "F"=Fahrenheit)

Alleen van toepassing op temperaturen, default is "C".

Returns:

Verwachte waarde van de opgegeven eigenschap over de opgegeven periode

Voorbeelden:

```
var = xxter.getmeteoinfo("temperature", 8, "F")
```

```
var = xxter.getmeteoinfo("rain", 24)
```

Voorbeeld 1 geeft de verwachte temperatuur in Fahrenheit over 8 uur.

Voorbeeld 2 geeft de verwachte neerslag over 24 uur.

Tabel 1: Mogelijke eigenschappen voor meteo functies

"temperature"	Temperatuur (in C, F of K)	
"feels like"	Gevoelstemperatuur (in C, F of K)	
"pressure"	Luchtdruk op zeeniveau (in hPa)	
"humidity"	Relatieve luchtvochtigheid (in %)	
"dew point"	Temperatuur van het dauwpunt bij condensatie (in C, F of K)	
"clouds"	Bewolking (in %)	
"visibility"	Gemiddelde zichtbaarheid (in meters)	NB: nvt voor min/max functies
"wind speed"	Windsnelheid (in m/s)	
"wind gust"	Windstoten, waar beschikbaar (in m/s)	
"wind degrees"	Windrichting, 0 = N, 90 = O, 180 = Z, 270 = W	NB: nvt voor min/max functies
"rain"	Verwachte hoeveelheid regen (in mm)	NB: nvt voor min functie
"snow"	Verwachte hoeveelheid sneeuw (in mm)	NB: nvt voor min functie
"precipitation"	Verwachte hoeveelheid neerslag (in mm)	NB: nvt voor min functie
"precipitation chance"	Kans op neerslag (regen en sneeuw)	NB: nvt voor min functie

```
xxter.getmeteoinfomin(what, from_offset, to_offset  
                      [, unit])
```

Beschrijving:

Deze functie geeft de minimum waarde terug van de opgegeven eigenschap in de opgegeven periode. Wanneer geen data beschikbaar is, of bij een fout, wordt *false* teruggegeven.

Parameters:

what : Eigenschap die moet worden opgevraagd, zie tabel 1 op de vorige pagina (string)
from_offset : Tijd in uren, vanaf wanneer de informatie moet zijn (number)
to_offset : Tijd in uren, tot wanneer de informatie moet zijn (number)
unit : Eenheid van de gevraagde gegevens (enum; "C"=Celsius, "K"=Kelvin, "F"=Fahrenheit)
Alleen van toepassing op temperaturen, default is "C".

Returns:

Verwachte minimale waarde van de opgegeven eigenschap gedurende de opgegeven periode

Voorbeelden:

```
var = xxter.getmeteoinfomin("temperature", 0, 8, "F")  
var = xxter.getmeteoinfomin("clouds", 24, 48)
```

Voorbeeld 1 geeft de minimale temperatuur in Fahrenheit tussen nu en over 8 uur.

Voorbeeld 2 geeft de minimale verwachte bewolking tussen over 24 en 48 uur.

```
xxter.getmeteoinfomax(what, from_offset, to_offset  
                     [, unit])
```

Beschrijving:

Deze functie geeft de maximum waarde terug van de opgegeven eigenschap in de opgegeven periode. Wanneer geen data beschikbaar is, of bij een fout, wordt *false* teruggegeven.

Parameters:

what : Eigenschap die moet worden opgevraagd, zie tabel 1 op de vorige pagina (string)
from_offset : Tijd in uren, vanaf wanneer de informatie moet zijn (number)
to_offset : Tijd in uren, tot wanneer de informatie moet zijn (number)
unit : Eenheid van de gevraagde gegevens (enum; "C"=Celsius, "K"=Kelvin, "F"=Fahrenheit)
Alleen van toepassing op temperaturen, default is "C".

Returns:

Verwachte maximale waarde van de opgegeven eigenschap gedurende de opgegeven periode

Voorbeelden:

```
var = xxter.getmeteoinfomax("temperature", 0, 8, "F")  
var = xxter.getmeteoinfomax("rain", 24, 48)
```

Voorbeeld 1 geeft de maximale temperatuur in Fahrenheit tussen nu en over 8 uur.

Voorbeeld 2 geeft de maximale verwachte neerslag tussen over 24 en 48 uur.

```
xxter.gettariff(delta_min)
```

Beschrijving:

Levert het actuele tarief voor het opgegeven tijdstip in de toekomst (in minuten), zoals ingesteld in de Smart Energy Manager. Wanneer een variabel tarief met spotprijzen is ingesteld, levert dit een dynamische waarde, anders levert dit het enkele of dubbele tarief dat geldt voor het opgegeven tijdstip.

Parameters:

delta_min : Het aantal minuten in de toekomst (nummer)

Returns:

Het actuele tarief, zoals ingesteld in de Smart Energy Manager.

Voorbeeld:

```
var = xxter.gettariff(60)
```

```
xxter.getusername()
```

Beschrijving:

Levert de gebruikersnaam van de gebruiker die de functie heeft gestart. Wanneer een (lokale) gebruiker een functie start, bijvoorbeeld door een trigger van een actie of vanuit de app, dan wordt de naam van deze gebruiker teruggegeven met deze functie. Wanneer het script op een andere manier wordt gestart, bijvoorbeeld als deze door de automatisering wordt getriggerd, dan wordt de naam van de hoofdgebruiker teruggegeven.

Parameters: -**Returns:**

De naam van de gebruiker die de functie heeft doen starten.

Voorbeeld:

```
var = xxter.getusername()
```

```
xxter.fadeto(duration_sec, component_id, value [,  
value 2, value 3] [, step_msec])
```

Beschrijving:

Verloopt de waarde van het component van de huidige stand naar de opgegeven waarde(n) gedurende de opgegeven tijd. De function komt direct terug, het verlopen gaat op de achtergrond.

Parameters:

duration_sec : Het aantal seconden dat het verlopen naar de nieuwe waarde duurt (nummer)

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)

step_msec : Aantal milliseconden van iedere stap (waardewijziging) in verloop (nummer)
optioneel. Default waarde is 500, input mag zijn tussen 250 en 60000.

Returns: -**Voorbeelden:**

`xxter.fadeto(30, 312, 100)` - verloopt comp. 312 naar waarde 100 in 30 sec., iedere 500 msec.

`xxter.fadeto(60, 75, 135, 230, 176, 250)` - verloopt comp. 75 naar RGB waarde 135,230,176 in 60 sec. iedere 250 msec.

Ondersteunde algemene Lua-functies

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

Tabel 2: ondersteunde standaard Lua-functies	
<pre> 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]]) </pre>	<pre> 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) </pre>

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