Manual de scripts Lua de xxter

Versión 1.7 – Abril 2025

Introducción a scripts de xxter

Con los scripts de xxter puede crear sus propios programas pequeños dentro de xxter. Estos scripts son muy flexibles y pueden utilizarse para añadir una amplia variedad de funciones a una instalación domótica en una vivienda o edificio. Puede crear funciones lógicas, retrasar acciones, ampliar sus escenarios con secuencias RGB y mucho más. Los scripts se pueden activar utilizando un escenario, una programación, una acción (desencadenante) u otro script.

xxter admite tanto un lenguaje de scripting nativo como Lua. Este manual proporciona explicaciones sobre los scripts Lua. Para los scripts nativos de xxter, consulte la página de documentación de nuestro sitio web (https://www.xxter.com/documentation/). También tenemos varios ejemplos disponibles en esta página y también en nuestro foro (https://forum.xxter.com).

Para obtener más información sobre Lua en general, consulte el manual de Lua: https://www.Lua.org/manual/5.3/

Lo básico

Los scripts xxter Lua son programas escritos que constan de una o más líneas. Puede añadir comentarios a un script escribiendo «--». Esto se puede hacer en una línea separada o al final de un mando.

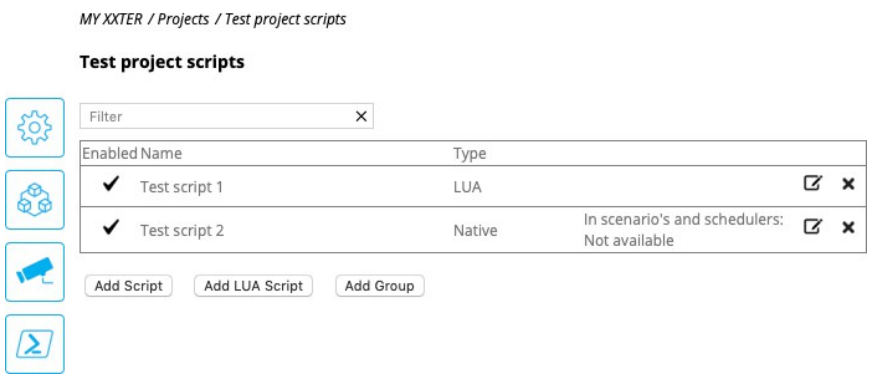
Un ejemplo de un script xxter Lua:

```
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
```

Los scripts se pueden iniciar y detener por muchos medios, por ejemplo, por un escenario o una programación. Los scripts pueden ser scripts «interminables» que realizan ciertas acciones en intervalos repetidos o pueden definirse como una secuencia de mandos que se ejecutan solo una vez, cada vez que se inicia el script. Al usar scripts «interminables» que se repiten continuamente, recuerde incluir siempre un período de «suspensión» para evitar que el script cree una carga muy pesada en el controlador xxter.

Gestión de scripts

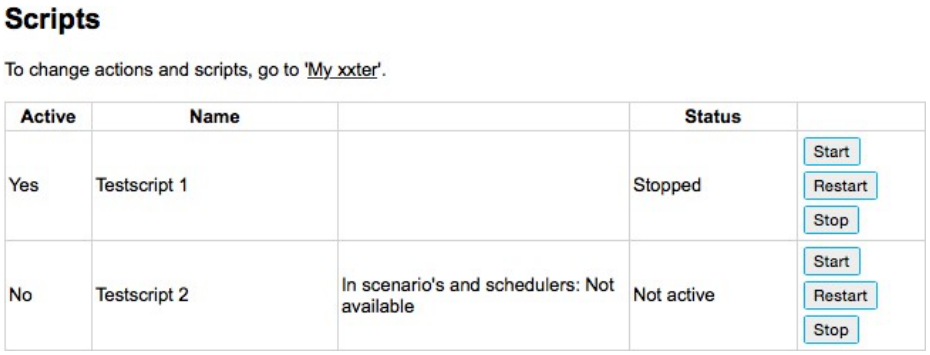
los scripts xxter se pueden crear y actualizar utilizando el editor xxter. Este editor se puede encontrar en la configuración del proyecto de «Mi xxter». Seleccione el proyecto para el que desea gestionar los scripts y seleccione «Scripts» en la barra de menú. Todos los scripts existentes se mostrarán aquí y se pueden editar y eliminar. Al crear un nuevo script, puede elegir si desea crear un script xxter nativo o un script Lua.



Importante: Cuando añade o edita un script, este debe cargarse en el controlador xxter antes de poder usarlo.

Cada vez que se ha realizado un cambio, puede ver el texto «Project has changed» en la esquina superior derecha. Al hacer clic en esto y posteriormente en «Push configuration to xxter device», el controlador descargará la nueva configuración. Esta función funciona mejor cuando xxter está configurado para su uso en exteriores (consulte el manual de instalación). Alternativamente, puede volver a cargar el perfil utilizando la aplicación (consulte el manual del usuario). Por supuesto, también puede volver a cargar el proyecto desde el propio controlador xxter, iniciando sesión en el controlador xxter y haciendo clic en el pulsador «Load configuration» en la esquina superior izquierda del menú.

Cuando haya iniciado sesión en el controlador xxter, puede ver los scripts cargados haciendo clic en la opción «Scripts» en el menú. Aquí también se puede iniciar, reiniciar o detener manualmente un script con fines de prueba.

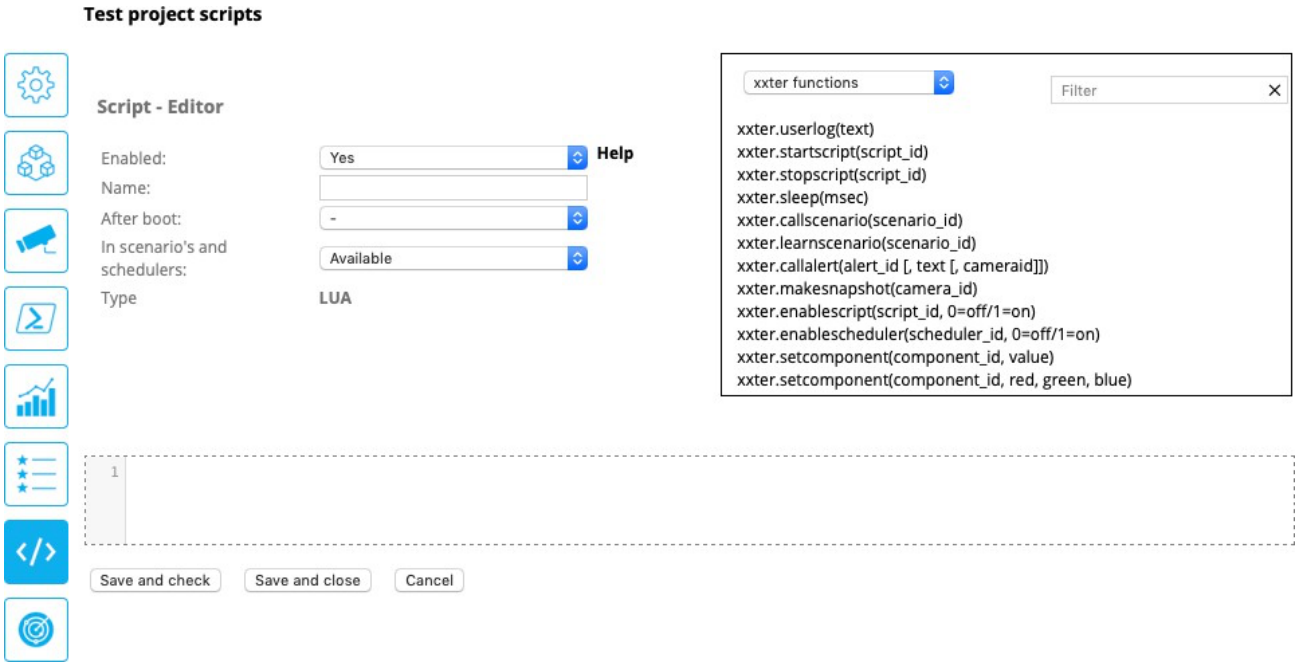


Para solucionar problemas con sus scripts, los registros del dispositivo son muy útiles. En el controlador xxter, puede habilitar «Scripts» en el inicio de sesión del usuario en la página *Settings – basic.* Esto registrará cada línea ejecutada de cada script en el registro del dispositivo. Puede acceder al registro a través de «Open user log window» en la página *de Status*.

Editor de scripts

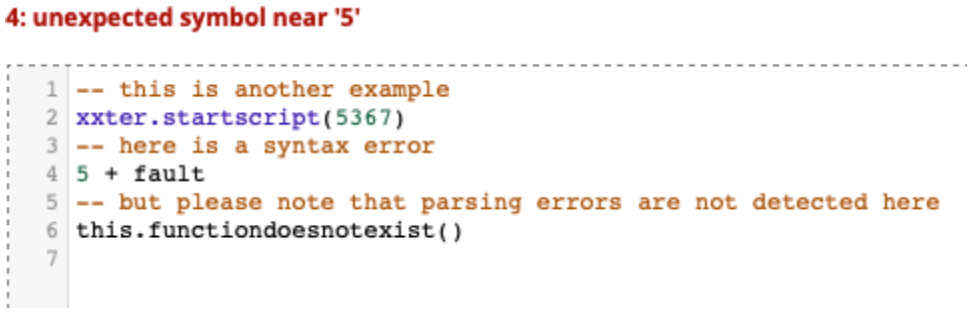
Cuando añada o edite un script, puede seleccionar si debe estar habilitado y puede seleccionar si el script debe estar disponible para el usuario final, para usar en escenarios o para el programador. También puede seleccionar si el script debe iniciarse automáticamente, siempre que se (re) inicie el controlador xxter.

IMPORTANTE: Los scripts deshabilitados no se pueden ejecutar y no se activarán cuando se incluyan en un escenario, acción o programador. Esto puede ser útil para pruebas y propuestas de solución de problemas. Sin embargo, ¡tenga en cuenta que un script puede ser habilitado o deshabilitado por otro script!

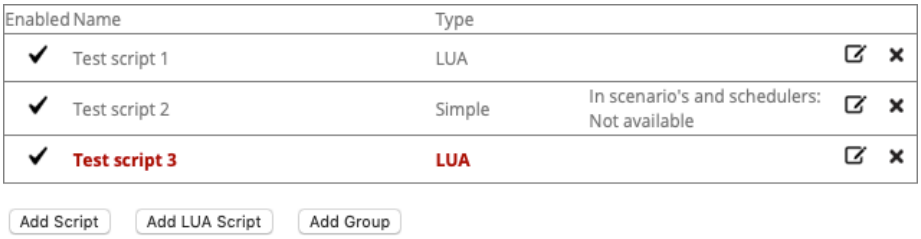


Los números de línea que se muestran junto al script en el editor son solo informativos y no se utilizan en los propios scripts. Los mandos se pueden añadir escribiéndolos directamente o utilizando la herramienta de selección de mandos, situada en la esquina superior derecha. Asegúrese de estar en la posición correcta en el editor de scripts al añadir un mando con la herramienta de mandos. Al añadir un mando con la herramienta mandos, las variables deben establecerse en los valores apropiados. Las funciones proporcionarán información sobre qué tipo de valores deben añadirse. Los ID de los componentes se pueden seleccionar desde la herramienta de selección, seleccionando el tipo de valor que necesita (por ejemplo, «scripts» o «floating points»). Al hacer clic en un componente, lo añadirá al script en la ubicación del cursor.

Después de haber añadido uno o más mandos en el editor, puede verificar si son válidos haciendo clic en el pulsador «Save and check». Su script actual se verificará y se volverá a mostrar en el editor. Si hay un error de sintaxis en el script, se mostrará un mensaje encima de la pantalla de edición. Sin embargo, los errores de análisis (por ejemplo, validar si las funciones existen o tienen la cantidad y el tipo de variables correctos) no se pueden detectar en el editor.



Después de corregir o añadir líneas, puede verificar fácilmente el script nuevamente utilizando el pulsador «Save and check». Cuando todo esté correcto puede utilizar el pulsador «Save and close» para cerrar el editor.
Los scripts que se guardan pero contienen un error se muestran en rojo.



Nota: los scripts que contienen errores se pueden guardar, pero no se iniciarán.

Los scripts dejarán de ejecutarse automáticamente después de que se haya ejecutado el último mando del script.

Los scripts se escriben utilizando funciones y valores. Además, los scripts se pueden extender utilizando variables y estructuras de control como sentencias IF y bucles WHILE. También es posible definir sus propias funciones, con parámetros de entrada y valores de retorno.

Para obtener más información sobre Lua, consulte el manual de Lua:
https://www.Lua.org/manual/5.3/

Funciones xxter

Las siguientes funciones específicas Lua de xxter están disponibles.

---

xxter.userlog(text)

Descripción:
Escribe en el registro de usuario, útil para depurar sus scripts Lua. La escritura en el registro de usuario de esta manera siempre tendrá lugar, incluso cuando el registro de Script se haya deshabilitado en la página *Settings – basic* del dispositivo, consulte la página 3 de este manual.

Parámetros:
text: una cadena con el texto a escribir en el registro de usuario.

Devoluciones: -

Ejemplo:
```
xxter.userlog («Ejecutando script parte X»)
```

---

xxter.startscript (script_id)

Descripción:
Inicia un script xxter, si aún no se está ejecutando.

Parámetros:
script_id: El identificador del script (número)

Devoluciones: -

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

---

xxter.stopscript(script_id)

Descripción:
Detiene un script xxter.

Parámetros:
script_id: El identificador del script (número)

Devoluciones: -

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

---

xxter.enablescript(script_id, status)

Descripción:
Activa o desactiva un script xxter, por lo que puede (no) iniciarse.

Parámetros:
script_id: El identificador del script (número)
estado: Estado del script deseado (enumeración: 0 = OFF, 1 = ON)

Devoluciones: -

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

---

xxter.include(script_id)

Descripción:
Incluye todo el código del script designado en el script actual. Esto, por ejemplo, le permite definir un conjunto de funciones en un script y usar ese mismo código en varios scripts.

Parámetros:
script_id: El identificador del script (número)

Devoluciones: -

Ejemplo:
```
xxter.include(17)
```

---

xxter.sleep(time)

Descripción:
Espera la cantidad de tiempo dada (en milisegundos), antes de continuar.

Parámetros:
tiempo: El periodo de tiempo en milisegundos para esperar (número)

Devoluciones: -

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

---

xxter.callscenario(scenario_id)

Descripción:
Llama al escenario xxter especificado para realizar sus acciones configuradas.

Parámetros:
id_escenario: El identificador del escenario (número)

Devoluciones: -

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

xxter.learnscenario(scenario_id)

Descripción:
Actualiza las acciones configuradas del escenario xxter especificado a sus valores actuales

Parámetros:
id_escenario: El identificador del escenario (número)

Devoluciones: -

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

---

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

Descripción:
Llama a la alerta xxter especificada, opcionalmente con texto específico e instantánea de cámara. El texto (opcional) que se proporciona se coloca en la posición «[x]» del servicio de alerta, si está disponible.

Parámetros:
alert_id: El identificador de la alerta (número)
texto: *Optional,* el texto que se enviará con la alerta (cadena)
camera_id: *Optional,* el identificador de la cámara para tomar una instantánea (número)

Devoluciones: -

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

---

xxter.makesnapshot(camera_id)

Descripción:
Hace una instantánea de la cámara proporcionada.

Parámetros:
camera_id: El identificador de la cámara para tomar una instantánea (número)

Devoluciones: -

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

---

xxter.enablescheduler(scedule_id, status)

Descripción:
Habilita o deshabilita un programador xxter, por lo que puede (no) iniciarse.

Parámetros:
scedule_id: El identificador del horario (número)
estado: Estado del programador deseado (enumeración: 0 = OFF, 1 = ON)

Devoluciones: -

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

---

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

Descripción:
Establece un componente con el valor proporcionado.

Parámetros:
id_componente: El identificador del componente (número)
valor: Valor deseado del componente (el tipo depende del componente).
En caso de que el componente sea RGB, este valor se refiere a R (número)
value_2: *Solo para RGB,,* se refiere a G (número)
value_3: *Solo para RGB,,* se refiere a B (número)

Devoluciones: -

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

---

xxter.getcomponent(component_id)

Descripción:
Obtiene el valor actual de un componente.

Parámetros:
id_componente: El identificador del componente (número)

Devoluciones:
valor: Valor actual del componente (el tipo depende del componente).
Componentes RGB, devuelven tres valores

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

---

xxter.readcomponent(component_id)

Descripción:
Ejecuta un mando de lectura en el bus KNX del componente. Si el componente KNX responde a la solicitud, xxter actualizará su estado actual (esto puede llevar algún tiempo). Posteriormente, el valor actual se puede utilizar en el script con *xxter.getcomponent*).

Parámetros:
id_componente: El identificador del componente (número)

Devoluciones: -

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

---

xxter.httpcommand(http_id [, result])

Descripción:
Ejecuta el mando HTTP proporcionado.

Parámetros:
http_id: El identificador del mando http (número)
resultado: Parámetro opcional para solicitar también el resultado del mando (booleano)

Devoluciones:
resultado: Si se solicita y está disponible, el resultado que devuelve el mando HTTP (cadena)

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

---

xxter.tcpcommand(tcp_id)

Descripción:
Ejecuta el mando TCP proporcionado.

Parámetros:
tcp_id: El identificador del mando tcp (número)

Devoluciones: -

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

---

xxter.ircommand(ir_id)

Descripción:
Ejecuta el mando de infrarrojos proporcionado.

Parámetros:
ir_id: El identificador del mando IR (número)

Devoluciones: -

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

---

xxter.openKNXtunnel(status)

Descripción:
Abre o cierra el túnel KNX.

Parámetros:
estado: Mando para abrir o cerrar el túnel (enum; 0=cerrar, 1=abrir)

Devoluciones: -

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

---

xxter.setsimulation(status)

Descripción:
Establece la simulación de presencia en el estado deseado.

Parámetros:
estado: Estado deseado de la simulación de presencia (enum; 0=parada, 1=registro, 2=reproducción)

Devoluciones: -

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

---

xxter.getsimulation()

Descripción:
Obtiene el estado actual de la simulación de

presencia. Parámetros: -

Devoluciones:
Estado actual de la simulación de presencia (enum; 0=parada, 1=registro, 2=reproducción)

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

---

xxter.connectKNX(estado)

Descripción:
Conecte o desconecte la conexión KNX del dispositivo.

Parámetros:
estado: Mando para la conexión KNX (enum; 0=desconectar, 1=conectar)

Devoluciones: -

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

---

xxter.setpersistent(varname, value)

Descripción:
Cree o actualice una variable que sea persistente en todos los scripts, para pasar información entre scripts.

Parámetros:
varname: Nombre de la variable persistente (cadena, alfanumérica: 0-9a-zA-Z)
valor: Valor que debe tener la variable persistente (tipo dependiendo de la variable)

Devoluciones: -

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

---

xxter.getpersistent(varname)

Descripción:
Recuperar una variable que sea persistente en todos los scripts, para pasar información entre scripts.

Parámetros:
varname: Nombre de la variable persistente (cadena, alfanumérica: 0-9a-zA-Z)

Devoluciones:
Valor de la variable persistente (tipo dependiendo de la variable)

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

---

xxter.getSunAzimuth()

Descripción:
Obtenga el ángulo de sol actual, en relación con el norte (dirección), en función de la ubicación configurada en el controlador xxter.

Parámetros: -

Devoluciones:
Valor de la dirección del sol.

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

---

xxter.getSunAltitude()

Descripción:
Obtenga el ángulo de sol actual, en relación con el horizonte (altitud), en función de la ubicación configurada en el controlador xxter.

Parámetros: -

Devoluciones:
Valor de la altitud del sol.

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

---

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

Descripción:
Ejecuta un mando de Sonos

Parámetros:
device_id: Se puede establecer en «global» para los mandos globales o para proporcionar un identificador de dispositivo Sonos específico (formato: RINCON_949F3EC192E401400).
command: cuando el identificador de dispositivo se establece en «global», se pueden utilizar los siguientes mandos: “creategroup”: requiere optional_setting “muteall” para el identificador de grupo (número)
"unmuteall"
"pauseall"
"playall"

cuando se establece el identificador de dispositivo de un dispositivo Sonos, se pueden utilizar los siguientes mandos: “groupvolume”: requiere optional_setting para el volumen (número) “groupmute”
"volumen": requiere optional_setting para el volumen (número)
“mute”, “unmute”
“play” , “pause”
“seek” : requiere optional_setting para la posición (número)

“next”, “previous”
"selectplaylist": requiere optional_setting para el nombre de la lista de reproducción (cadena) "selectfavorite": requiere optional_setting para el nombre del favorito (cadena) "selectnextfavorite"
“selectlinein”
“selectHDMI”
“playaudioclip”: requiere dos optional_setting, uno para el tipo de clip de audio (enum) y otro para el número de clip (número). Los tipos de clips de audio son:
0: Estándar Sonos
1: estándar xxter
2: personalizado

Optional_setting: dependiendo del mando utilizado (consulte los parámetros del mando anteriores)

Devoluciones: -

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

Los ejemplos 1 y 2 son mandos globales, para crear un grupo de altavoces o silenciar a todos los altavoces. Los ejemplos 3 y 4 son para un altavoz, para reproducir "Radio 4" o reproducir un clip de audio personalizado.

---

xxter.getsonos(device_id, parameter)

Descripción:
Solicitar el estado actual de un dispositivo Sonos

Parámetros:
device_id: Identificador específico del dispositivo Sonos (formato: RINCON_949F3EC192E401400) parameter: El parámetro que se va a devolver (seleccione uno de los siguientes:
“groupvolume”, “volume”, “status”, “mute”, “groupmute”, “meta”)

Devoluciones:
groupvolume: volumen actual del grupo Sonos del dispositivo (número)
volume: volumen actual del dispositivo Sonos (número)
status: estado actual del dispositivo Sonos (enum: 0:IDLE, 1:BUFFER, 2:PLAY, 3:PAUSED)
(group)mute: estado de silenciamiento actual (grupo) (1 si está silenciado, 0 si no está silenciado)
a, b, c, d, e, f, g: metadatos del título de reproducción actual (varias cadenas)
A = track_name
B = track_artist
C = nombre_álbum
D = album_artist
E = show_name
F = container_name
G = stream_info

Ejemplos:
```
groupvol = xxter.getsonos("RINCON_949F3EC192E401400", "groupvolume")
a, b, c, d, e, f, g = xxter.getsonos("RINCON_949F3EC192E401400", "meta")
```
El ejemplo 1 solicita el volumen de grupo, el ejemplo 2 los metadatos de la pista actual.

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

Descripción:
Ejecuta un mando uPnP

Parámetros:
device_id: Identificador específico del dispositivo uPnP (formato: RINCON_949F3EC192E401400) command: se pueden utilizar los siguientes:

"volumen":	requiere optional_setting para el volumen (número)
"mute"	
"unmute"	
"play"	
"pause"	
"seek":	requiere option_setting para la posición (número)
"next"	
"previous"	

optional_setting: dependiendo del mando utilizado (consulte los parámetros del mando anteriores)

Devoluciones: -

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

El ejemplo 1 establece el volumen del altavoz, el ejemplo 2 pone el altavoz en silencio.

---

xxter.getupnp(id_dispositivo)

Descripción:
Solicitar el estado actual de un dispositivo uPnP

Parámetros:
device_id: Identificador específico del dispositivo uPnP (formato: RINCON_949F3EC192E401400) parameter: El parámetro que se va a devolver (seleccione uno de: “volume”, “status”, “meta”)

Devoluciones:
volumen: volumen actual del dispositivo uPnP (número)
status: estado actual del dispositivo uPnP (cadena)
a, b, c, d, e, f, g: metadatos del título de reproducción actual (varias cadenas)

A = track_name
B = track_artist
C = nombre_álbum
D = album_artist
E = show_name
F = container_name
G = stream_info

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

El ejemplo 1 solicita el volumen del altavoz, el ejemplo 2 los metadatos de la pista actual.

xxter.timezonediff()

Descripción:
Proporciona la diferencia en segundos entre la zona horaria local y UTC (GMT sin horario de verano).

Parámetros: -

Devoluciones:
Diferencia de zona horaria con UTC en segundos

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

---

xxter.localtime()

Descripción:
Proporciona la hora local en segundos. Esto funciona de manera similar aos.clock () pero para la zona horaria local en lugar de UTC (GMT sin horario de verano). La función se puede utilizar, por ejemplo, junto con la función Luaos.date (formato, hora) como variable de hora.

Parámetros: -

Devoluciones:
Hora local en segundos

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

---

xxter.getmeteostatus()

Descripción:
Indica el número de horas transcurridas desde la última actualización de la información meteorológica. La información meteorológica se actualiza automáticamente, pero esta función se puede utilizar para saber cuándo se hizo esto por última vez.

Parámetros: -

Devoluciones:
Número de horas desde la última actualización meteorológica

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

---

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

Descripción:
Esta función devuelve el valor de la propiedad solicitada. En caso de que los datos no estén disponibles, se devuelve *false* .

Parámetros:
qué: Propiedad que debe devolverse, consulte la tabla 1 a continuación (cadena)
desplazamiento: Offset en horas, para indicar cuánto tiempo en el futuro debe estar la información (número)
Por ejemplo, «4» proporciona la información meteorológica esperada en 4 horas. El valor predeterminado es 0. unidad: Unidad del valor solicitado (enum; «C»=Celsius, «K»=Kelvin, «F»=Fahrenheit)
Solo se aplica a las temperaturas, el valor predeterminado es «C».

Devoluciones:
Valor esperado de la propiedad proporcionada en el período de tiempo proporcionado

Ejemplos:
```
var = xxter.getmeteoinfo("temperature", 8, "F")
var = xxter.getmeteoinfo("rain", 24)
```

El Ejemplo 1 da la temperatura esperada en Fahrenheit en 8 horas. El ejemplo 2 da la precipitación que se espera en 24 horas.

Tabla 1: Posibles propiedades para las funciones meteo	
«temperatura»	Temperatura (en C, F o K)
«feels like»	Temperatura por percepción humana (en C, F o K)
«pressure»	Presión atmosférica a nivel del mar (en hPa)
«humidity»	Humedad relativa del aire (en %)
«dew point»	Temperatura a la que las gotas de agua comienzan a condensarse (en C, F o K)
«uv index»	Índice de riesgo de radiación ultravioleta
«nubes»	Cobertura en la nube (en %)
«visibility»	Visibilidad media (en metros) **- n/d con funciones mín/máx**
«wind speed»	Velocidad del viento (en m/s)
«wind gust»	Ráfagas de viento, cuando estén disponibles (en m/s)
«wind degrees»	Dirección del viento, 0 = N, 90 = E, 180 = S, 270 = W **- n/d con funciones mín/máx**
«lluvia»	Cantidad esperada de lluvia (en mm) **- n/d con función mín**
«snow»	Cantidad esperada de nieve (en mm) **- n/d con función min**
«precipitation»	Cantidad esperada de precipitación (en mm) **- n/d con función min**
«probabilidad de precipitación»	Posibilidad de precipitación (lluvia o nieve) **- n/d con función min**

xxter.getmeteoinfomin(what, from_offset, to_offset [, unidad])

Descripción:
Esta función devuelve el valor mínimo de la propiedad solicitada durante el período de tiempo proporcionado. En caso de que los datos no estén disponibles, se devuelve *false* .

Parámetros:
qué: Propiedad que debe devolverse, consulte la tabla 1 en la página anterior (cadena) from_offset: Tiempo en horas, desde cuando la previsión debe ser (número)
to_offset: Tiempo en horas, hasta cuando la previsión debe ser (número)
unidad: Unidad del valor solicitado (enum; «C»=Celsius, «K»=Kelvin, «F»=Fahrenheit) Solo se aplica a las temperaturas, el valor predeterminado es «C».

Devoluciones:
Valor mínimo esperado de la propiedad proporcionada durante el período de tiempo proporcionado

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

El Ejemplo 1 da la temperatura mínima en grados Fahrenheit entre ahora y las próximas 8 horas. El ejemplo 2 proporciona la cobertura de nube mínima esperada entre 24 y 48 horas a partir de ahora.

---

xxter.getmeteoinfomax(what, from_offset, to_offset [, unidad])

Descripción:
Esta función devuelve el valor máximo de la propiedad solicitada durante el período de tiempo proporcionado. En caso de que los datos no estén disponibles, se devuelve *false* .

Parámetros:
qué: Propiedad que debe devolverse, consulte la tabla 1 en la página anterior (cadena) from_offset: Tiempo en horas, desde cuando la previsión debe ser (número)
to_offset: Tiempo en horas, hasta cuando la previsión debe ser (número)
unidad: Unidad del valor solicitado (enum; «C»=Celsius, «K»=Kelvin, «F»=Fahrenheit) Solo se aplica a las temperaturas, el valor predeterminado es «C».

Devoluciones:
Valor máximo esperado de la propiedad proporcionada durante el período de tiempo proporcionado

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

El Ejemplo 1 da la temperatura máxima en grados Fahrenheit entre ahora y las próximas 8 horas. El ejemplo 2 da la lluvia máxima esperada entre 24 y 48 horas a partir de ahora.

---

xxter.gettariff(delta_min)

Descripción:
Obtiene la tarifa real para el momento proporcionado en el futuro (en minutos), según lo establecido para el Smart Energy Manager. Si se configura una tarifa variable este será un valor dinámico, en caso contrario devolverá la tarifa única o fraccionada aplicable para ese momento.

Parámetros:
delta_min: La cantidad de minutos en el futuro (número)

Devoluciones:
La tarifa real, según lo establecido para el Smart Energy Manager.

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

---

xxter.getusername()

Descripción:
Obtiene el nombre de usuario del usuario que inició la función. Cuando un usuario (local) activa el script, por ejemplo, a través de una acción o la aplicación, se devolverá su nombre de usuario con esta función. Si el script se activa a través de cualquier otro medio, por ejemplo, a partir de un valor en la automatización, se devolverá el usuario principal.

Parámetros: -

Devoluciones:
El nombre del usuario que inició la función.

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

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

Descripción:
Hace que el componente pase gradualmente de su configuración actual al valor o valores indicados durante el tiempo especificado. La función volverá de inmediato y la atenuación se realizará en segundo plano.

Parámetros:

duration_sec:	Cantidad de segundos para realizar la transición al nuevo valor (número)
component_id:	El identificador del componente (número)
value:	Valor deseado del componente (el tipo depende del componente). En caso de que el componente sea RGB, este valor se refiere a R (número)
value_2:	*Solo para RGB,* se refiere a G (número)
value_3:	*Solo para RGB,* se refiere a B (número)
step_msec:	*Miliseg. para cada paso (cambio de valor) en la atenuación (número) opcional. El valor predeterminado es 500, la entrada puede estar entre 250 y 60000.*

Devoluciones: -

Ejemplo:
`xxter.fadeto(30,312,100)` - *atenuará el comp. 312 hasta el valor 100 en 30 segundos, con un paso cada 500 ms.*
`xxter.fadeto(60,75,135,230,176,250)`
-atenuará el componente 75 hasta el valor RGB 135,230,176 en 60 segundos, con un paso cada 250 ms.

Funciones generales de Lua admitidas

Además de estas funciones xxter específicas, también se admiten las siguientes funciones Lua estándar:

Tabla 2: funciones Lua estándar compatibles

```
tonumber (e [, base])
tostring (v)
tipo (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)
```

La documentación para estas funciones se puede encontrar en el manual de Lua:
https://www.Lua.org/manual/5.3/