Home Automation Recipes

Table of Contents

Light triggering

Turn on lights when motion is detected between sunset and sunrise

```
rule([[motion:breached & sunset..sunrise =>
  hallwayLight:on;
```

```
  log('Hallway light turned on due to motion')
]])
```

Turn on lights when motion is detected and fibaro global variable 'Vacation' is not true

```
rule([[motion:breached & !Vacation =>
  hallwayLight:on;
  log('Hallway light turned on due to motion')
]])
```

Turn on lights when scene activation event from switch

```
rule([[switch:scene == S1.double => -- double click
  hallwayLight:on;
  log('Double click switch, Hallway light turned on')
]])
```

Turn on lights when key 2 is pressed on remote control

```
rule([[remote:central.keyId == 2 =>
  hallwayLight:on;
  log('Remote key 2, Hallway light turned on')
]])
```

Scheduling

Set a global variable with day state

```
rule("@00:00 => weekDay = wday('mon-fri')").star()
rule("@00:00 => weekEnd = wday('sat-sun')").star()

rule("weekDay & 07:00..07:30 => $HomeState='WakeUp'").start()
rule("weekDay & 07:30..11:00 => $HomeState='Morning'").start()
rule("weekDay & 11:00..13:00 => $HomeState='Lunch'").start()
rule("weekDay & 13:00..18:30 => $HomeState='Afternoon')").start()
rule("weekDay & 18:30..20:00 => $HomeState='Dinner'").start()
rule("weekDay & 20:00..23:00 => $HomeState='Evening'").start()
rule("weekDay & 23:00..07:00 => $HomeState='Night'").start()

rule("weekEnd & 08:00..09:00 => $HomeState='WakeUp'").start()
rule("weekEnd & 09:00..12:00 => $HomeState='Morning'").start()
rule("weekEnd & 12:00..14:00 => $HomeState='Lunch'").start()
rule("weekEnd & 14:00..19:00 => $HomeState='Afternoon'").start()
rule("weekEnd & 19:00..20:00 => $HomeState='Dinner'").start()
rule("weekEnd & 20:00..24:00 => $HomeState='Evening'").start()
rule("weekEnd & 24:00..08:00 => $HomeState='Night'").start()
```

```
rule("@dawn+00:15 => $isDark=false")
rule("@dusk-00:15 => $isDark=true")
```

The 07:00..07:30 rule will trigger at 07:00 and 07:00:01 and check the condition. If the current time is between (inclusive) 07:00..07:30 we will set the global variable 'HomeState' to 'Wakeup'. The .start() added to the rule makes it run at startup, setting the variable correctly if it's between the times specified. Why the rule triggers on 07:00:01 is a technicality, needed if we negate the test, and usually nothing to be concerned of as it normally will be false anyway. WIth these variables a rule to turn on light if dark wakeup could be

```
rule("$HomeState=='WakeUp' & isDark => bedroomLight:on")
```

Turn off all lights at midnight

```
rule([[@00:00 =>
  allLights:off;
  log('All lights turned off at midnight')
]])
```

Turn off all lights at 23 on weekdays and midnight on weekends

```
rule([[23:00 & wday('mon-thu') =>
  allLights:off;
  log('All lights turned off at 23:00')
]])

rule([[@00:00 & wday('fri-sun') =>
  allLights:off;
  log('All lights turned off at midnight')
]])
```

Turn off lights on Earth Hour

```
rule("earthLight = {kitchen.lamp, bedroom.lamp}")
rule("log('Earth light IDs: %s',json.encodeFast(earthLight))")
rule("earthLight:on")

rule([[earthDates={
    2025/03/29/20:30,
    2026/03/28/20:30,
    2027/03/27/20:30
}]])

rule([[for _,t in ipairs(earthDates) do
      if t > os.time() then
        print('Earth hour date:',os.date('%c',t));
```

```
        post(#earthHour,t)
      end
    end
]])

rule([[#earthHour =>
  local state = {};
  log('Earth hour started');
  for _,id in ipairs(earthLight) do state[id] = id:value end;
  earthLight:off;
  wait(01:00);
  log('Earth hour ended');
  for id,val in pairs(state) do id:value=val end
]])
```

Restart EventRunner at Daylight Time Savings

```
rule("post(#restart,nextDST())")     -- post event at next daylight
rule("#restart => plugin.restart()") -- Restart QA when DST hour jum
```

This because rules like

```
rule("@15:00 => ...")
```

will be off an hour at DST day. The reason is that the time is calculated every midnight, and this is calculated as 15*3600 seconds after midnight. But at DST the real 15:00 jumps an hour forward or backward. It actually goes for all setTimeout set by the system, if the delay is calculated to run on an absolute time. So, the simplest approach is to just restart the QA at DST. All rules start up again and all timers are set with the new right time...

Security routines

Arm security system at night

```
rule([[@23:00 =>
  securitySystem:arm;
  log('Security system armed for the night')
]])
```

Disarm security system in the morning

```
rule([[@06:00 =>
  securitySystem:disarm;
  log('Security system disarmed for the day')
]])
```

Climate control

Turn on fan if temperature is high

```
rule([[temp:value > 28 =>
  fan:on;
  log('Fan turned on due to high temperature')
]])
```

Turn on fan if temperature is high for more than 5 min, and off when low for 5min

```
rule([[trueFor(00:05,temp:value > 28) =>
  fan:on;
  log('Fan turned on due to high temperature')
]])

rule([[trueFor(00:05,temp:value < 20) =>
  fan:on;
  log('Fan turned off due to low temperature')
]])
```

Notification examples

Send notification if door is left open for more than 5 minutes

```
rule("user = 456") -- Id of user that should be pushed to
rule([[trueFor(00:05,door:open) =>
  user:msg = log('Door open for %s minutes',5*again(10))
]])
```

Notification on last Monday in week

```
rule([[@18:00 & day('lastw-last') & wday('mon') =>
  user:msg = log('Last Monday in week, put out the trash')
]])
```