TellStick ZNet Documentation

Release v1.0.11

Telldus Technologies

Apr 28, 2017

Contents

TellStick ZNet allows developers to build own plugins and scripts run the device to extend the functionality with features not yet supported by Telldus.

It is also possible to alter the behaviour on how TellStick ZNet should interpret signals and messages from devices.

CHAPTER 1

Intro

TellStick ZNet offers two ways of integrating custom scripts. They can be written in either Python or Lua. The difference is outlined below.

Python

Python plugins are only available for TellStick ZNet Pro. Python plugins cannot be run on TellStick ZNet Lite. Python plugins offers the most flexible solution since full access to the service is exposed. This also makes it fragile since Python plugins can affect the service negative.

Lua

Lua code is available on both TellStick ZNet Pro and TellStick ZNet Lite. Lua code runs in a sandbox and has only limited access to the system.

To create a Lua script you need to access the local web server in TellStick ZNet. Browse to: http://{[}ipaddress{]}/lua to access the editor.

Globally accessible objects: deviceManager

Lua codes works by signals from the server triggers the execution.

CHAPTER 2

Lua

Example: Real wind

A thermometer measures the actual temperature but it is not the same as the perceived temperature. To get perceived temperature you must also take the wind into account. If TellStick ZNet has an anemometer this can be used to calculate the perceived temperature.

The script below calculates this and gives the anemometer a thermometer value.

Source of the algorithm: http://www.smhi.se/kunskapsbanken/meteorologi/vindens-kyleffekt-1.259

```
-- EDIT THESE

local windSensor = 287
local tempSensor = 297

-- DO NOT EDIT BELOW THIS LINE

local tempValue = deviceManager:device(tempSensor).sensorValue(1, 0)
local windValue = deviceManager:device(windSensor).sensorValue(64, 0)

function calculate()
  if tempValue == nil or windValue == nil then
    return
  end
  local w = math.pow(windValue, 0.16)
  local v = 13.12 + 0.6215*tempValue - 13.956*w + 0.48669*tempValue*w
  v = math.floor(v * 10 + 0.5) / 10
  local windDevice = deviceManager:device(windSensor)
  windDevice:setSensorValue(1, v, 0)
end

function onSensorValueUpdated(device, valueType, value, scale)
  if device:id() == windSensor and valueType == 64 and scale == 0 then
    windValue = value
```

```
    calculate()
  elseif device:id() == tempSensor and valueType == 1 and scale == 0 then
    tempValue = value
    calculate()
  end
end
```

Example: Zipato RFID

Telldus does not support the RFID reader from Zipato.

http://www.zipato.com/default.aspx?id=24&pid=88&page=1&grupe=0,2_15,3_37,0

It can be used any way with some Lua code.

```
-- Change these
local zipatoNodeId = 892

local tags = {}
-- Add tags below

-- Example code from a tag
-- tags[1] = {device=881, code={143, 188, 119, 84, 42, 0, 1, 4, 0, 0}};
-- Code for entering 1-2-3-4 on the keyboard
-- tags[2] = {device=813, code={49, 50, 51, 52, 0, 0, 0, 0, 0, 0}};

-- Do not change below

COMMAND_CLASS_USER_CODE = 0x63
USER_CODE_SET = 0x01
USER_CODE_REPORT = 0x03
COMMAND_CLASS_ALARM = 0x71
ALARM_REPORT = 0x05

local zipatoNode = deviceManager:device(zipatoNodeId):zwaveNode()

function compareTags(tag1, tag2)
      for index, item in python.enumerate(tag2) do
              if item ~= tag1[index+1] then
                      return false
              end
      end
      return true
end

function configureTag(index)
      local data = list.new(index, 1)
      for key,code in pairs(tags[index]['code']) do
              data.append(code)
      end
      zipatoNode:sendMsg(COMMAND_CLASS_USER_CODE, USER_CODE_SET, data)
      print("A new tag was configured in the Zipato.")
      print("This will be sent the next time the reader is awake")
end

function checkNewTag(code)
```

```
	-- New tag received. Check if it should be configured?
	for key,tag in pairs(tags) do
		if compareTags(tag['code'], code) then
			configureTag(key)
			return
		end
	end
	-- Not yet configured. Must be configured first.
	print("New unknown tag received. Add this to the codes if this should be␣
↪recognized")
	print("Tag data is %s", code)
end

function handleAlarm(data)
	if list.len(data) < 8 then
		return
	end

	local event = data[5]
	local tag = data[7]
	local device = deviceManager:device(tags[tag]['device'])
	if device == nil then
		print("Device not found")
	end
	if event == 5 then
		print("Away, tag %s", tag)
		zipatoNode:sendMsg(0x20, 0x01, list.new(0xFF))
		device:command("turnoff", nil, "RFID")
	elseif event == 6 then
		print("Home, tag %s", tag)
		device:command("turnon", nil, "RFID")
	end
end

function onZwaveMessageReceived(device, flags, cmdClass, cmd, data)
	if device:id() ~= zipatoNodeId then
		return
	end
	if cmdClass == COMMAND_CLASS_ALARM and cmd == ALARM_REPORT then
		handleAlarm(data)
		return
	end
	if cmdClass ~= COMMAND_CLASS_USER_CODE or cmd ~= USER_CODE_REPORT then
		return
	end
	local identifier = data[0]
	local status = data[1]
	if identifier == 0 and status == 0 then
		checkNewTag(list.slice(data,2))
		return
	end
end

-- This command clears all configured codes in the reader
-- zipatoNode:sendMsg(COMMAND_CLASS_USER_CODE, USER_CODE_SET, list.new(0, 0))
```

CHAPTER 3

Local API

TellStick ZNet has a local REST interface to integrate into third party applications not running on the TellStick ZNet itself

A list of all available functions can be browsed on the device itself. Browse to: http://{[}ipaddress{]}/api to list the functions.

Authentication

Before making any REST calls to TellStick ZNet the application must request a token that the user has authenticated.

Step 1 - Request a request token

Request a request token by performing a PUT call to the endpoint /api/token. You need to supply the application name as a parameter "app"

```
$ curl -i -d app="Example app" -X PUT http://0.0.0.0/api/token
HTTP/1.1 200 OK
Date: Fri, 15 Jan 2016 13:33:54 GMT
Content-Length: 148
Content-Type: text/html;charset=utf-8
Server: CherryPy/3.8.0

{
  "authUrl": "http://0.0.0.0/api/authorize?token=0996b21ee3f74d2b99568d8207a8add9",
  "token": "0996b21ee3f74d2b99568d8207a8add9"
}
```

Step 2 - Authenticate the app

Redirect the user to the url returned in step 1 to let him/her authenticate the app.

Step 3 - Exchange the request token for an access token

When the user has authenticated the request token in step 2 the application needs to exchange this for an access token. The access token can be used in the API requests. To exchange the request token for an access token perform a GET call to the same endpoint in step 1. Supply the request token as the parameter "token".

```
$ curl -i -X GET http://0.0.0.0/api/token?token=0996b21ee3f74d2b99568d8207a8add9
HTTP/1.1 200 OK
Date: Fri, 15 Jan 2016 13:39:22 GMT
Content-Length: 230
Content-Type: text/html;charset=utf-8
Server: CherryPy/3.8.0

{
  "allowRenew": true,
  "expires": 1452951562,
  "token":
↪"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImF1ZCI6IkV4YW1wbGUgYXBwIiwiZXhwIjoxNDUyOTUxNTYyfQ.
↪eyJyZW5ldyI6dHJ1ZSwidHRsIjo4NjQwMH0.HeqoFM6-K5IuQa08Zr9HM9V2TKGRI9VxXlgdsutP7sg"
}
```

If the returned data contains allowRenew=true then the token was authorized to renew its expiration itself without letting the user authorize the app again. The application must renew the token before it expires or else the application must start the autorization again from step 1.

If allowRenew is not set to true it is not possible for the app to renew the token and it will always expire after the time set in the parameter "expires".

Step 4 - Making a request

To make a requst to a TellStick ZNet API endpoint the token in step 3 must be supplied as a bearer token in the header. This is an example requesting a list of devices:

```
$ curl -i -X GET http://0.0.0.0/api/devices/list?supportedMethods=3 -H
↪"Authorization: Bearer␣
↪eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImF1ZCI6IkV4YW1wbGUgYXBwIiwiZXhwIjoxNDUyOTUxNTYyfQ.
↪eyJyZW5ldyI6dHJ1ZSwidHRsIjo4NjQwMH0.HeqoFM6-K5IuQa08Zr9HM9V2TKGRI9VxXlgdsutP7sg"
HTTP/1.1 200 OK
Date: Tue, 19 Jan 2016 10:21:29 GMT
Content-Type: Content-Type: application/json; charset=utf-8
Server: CherryPy/3.7.0

{
  "device": [
    {
      "id": 1,
      "methods": 3,
      "name": "Example device",
      "state": 2,
      "statevalue": "",
      "type": "device"
    }
  ]
}
```

Refreshing a token

If the user allowed the application to renew the token in steg 2 it can be renewed by the calling application. The token must be refreshed before it expires. If the token has expired the authentication must be restarted from step 1 again.

```
$ curl -i -X GET http://0.0.0.0/api/refreshToken -H "Authorization: Bearer␣
↪eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImF1ZCI6IkV4YW1wbGUgYXBwIiwiZXhwIjoxNDUyOTUxNTYyfQ.
↪eyJyZW5ldyI6dHJ1ZSwidHRsIjo4NjQwMH0.HeqoFM6-K5IuQa08Zr9HM9V2TKGRI9VxXlgdsutP7sg"
HTTP/1.1 200 OK
Date: Tue, 19 Jan 2016 10:21:29 GMT
Content-Type: Content-Type: application/json; charset=utf-8
Server: CherryPy/3.7.0

{
  "expires": 1455295348,
  "token":
↪"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImF1ZCI6IkV4YW1wbGUgYXBwIiwiZXhwIjoxNDU1Mjk1MzQ4fQ.
↪eyJyZW5ldyI6dHJ1ZSwidHRsIjo4NjQwMH0.M4il4_2SqJwlCjmuXlU5DS6h-gX7493Tnk9oBJXbgPw"
}
```

The new token returned must be used from now on and the old be discarded.

CHAPTER 4

API reference

Module: base

Classes in the base module are only accessible from Python applications.

Module: scheduler

Module: telldus