



Intermediate

# Philips Hue lamps controlled from Wiser for KNX

In Wiser for KNX

## Application note

Generation 2

LSS100100

10/2018

rev. 1



# Legal Information

The Schneider Electric brand and any registered trademarks of Schneider Electric Industries SAS referred to in this guide are the sole property of Schneider Electric SA and its subsidiaries. They may not be used for any purpose without the owner's permission, given in writing. This guide and its content are protected, within the meaning of the French intellectual property code (Code de la propriété intellectuelle français, referred to hereafter as "the Code"), under the laws of copyright covering texts, drawings and models, as well as by trademark law. You agree not to reproduce, other than for your own personal, noncommercial use as defined in the Code, all or part of this guide on any medium whatsoever without Schneider Electric's permission, given in writing. You also agree not to establish any hypertext links to this guide or its content. Schneider Electric does not grant any right or license for the personal and noncommercial use of the guide or its content, except for a non-exclusive license to consult it on an "as is" basis, at your own risk. All other rights are reserved.

Electrical equipment should be installed, operated, serviced and maintained only by qualified personnel. No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this material.

As standards, specifications and designs change from time to time, please ask for confirmation of the information given in this publication.

## Trademarks

- Microsoft Windows®, Windows 7®, Windows 10® and Internet Explorer® are trademarks or registered trademarks of the Microsoft Corporation in the USA and/or other countries.
- iTunes™ is a registered or unregistered trademark of the Apple Inc. in the USA and/or other countries.
- Google Chrome™, Google Play™, Google Maps™ and YouTube™ are trademarks or registered or unregistered trademarks of the Google Inc. in the USA and/or other countries.
- Firefox® is a trademark or registered trademark of the Mozilla Corporation in the USA and/or other countries.

# Important Safety Information

Read these instructions carefully and look at the equipment to become familiar with the device before trying to install, operate, service, or maintain it. The following special messages may appear throughout this manual or on the equipment to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.



The addition of either symbol to a “Danger” or “Warning” safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.



This is the safety alert symbol. It is used to alert you to potential personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

## **DANGER**

**DANGER** indicates a hazardous situation which, if not avoided, **will result in** death or serious injury.

## **WARNING**

**WARNING** indicates a hazardous situation which, if not avoided, **could result in** death or serious injury.

## **CAUTION**

**CAUTION** indicates a hazardous situation which, if not avoided, **could result in** minor or moderate injury.

## **NOTICE**

NOTICE is used to address practices not related to physical injury.

# Table of Contents

1	Introduction	8
1.1	Added value of this application note	8
1.2	Competencies	8
1.3	System prerequisites	8
2	Design	9
3	Basic user.hue library description	10
4	Configuration	12
4.1	Hue ZigBee Bridge Setup	12
4.2	Wiser for KNX and Hue ZigBee Bridge setup	12
5	Hue control	14
5.1	Check user existence on the bridge	14
5.2	Delete (unlink) the user from the bridge	14
5.3	Turn the lamp on with 50% brightness	15
5.4	Turn the lamp off	15
5.5	Set lamp color in xy color space	15
5.6	Turn the lamp on with brightness, saturation and hue color	15
5.7	Set color temperature	15
5.8	Set color using RGB object	15
5.9	Make all lights blinking for 30s	15
5.10	Put all lights into color loop	15
5.11	API dot notation	16
6	Controlling the lights state	17
6.1	Light state body arguments	17
6.2	XY coordinates in CIE color space	18
6.3	Hue color + saturation	18
6.4	White color temperature	18
6.5	RGB color setting	18
7	Sample project	20
8	Conclusion	21
9	Appendix	22
9.1	Glossary	22
9.2	Reference	22



## Please note

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this material.

A qualified person is one who has skills and knowledge related to the construction, installation, and operation of electrical equipment and has received safety training to recognize and avoid the hazards involved.

## Safety Precautions

### **WARNING**

#### **HAZARD OF INCORRECT INFORMATION**

- Do not incorrectly configure the software, as this can lead to incorrect reports and/or data results.
- Do not base your maintenance or service actions solely on messages and information displayed by the software.
- Do not rely solely on software messages and reports to determine if the system is functioning correctly or meeting all applicable standards and requirements.
- Consider the implications of unanticipated transmission delays or failures of communications links.

**Failure to follow these instructions can result in death, serious injury, or equipment damage.**



**Attention** - the information provided must be complied with, otherwise program or data errors may occur.



**Note** - You will find additional information here to make your work easier.

The information provided in this documentation contains general descriptions and/or technical characteristics of the performance of the products contained herein. This documentation is not intended as a substitute for and is not to be used for determining suitability or reliability of these products for specific user applications. It is the duty of any such user or integrator to perform the appropriate and complete risk analysis, evaluation and testing of the products with respect to the relevant specific application or use thereof. Neither Schneider Electric nor any of its affiliates or subsidiaries shall be responsible or liable for misuse of the information that is contained herein. If you have any suggestions for improvements or amendments or have found errors in this publication, please notify us.

No part of this document may be reproduced in any form or by any means, electronic or mechanical, including photocopying, without express written permission of Schneider Electric.

All pertinent state, regional, and local safety regulations must be observed when installing and using this product. For reasons of safety and to help ensure compliance with documented system data, only the manufacturer should perform repairs to components.

When devices are used for applications with technical safety requirements, the relevant instructions must be followed.

Failure to use Schneider Electric software or approved software with our hardware products may result in injury, harm, or improper operating results.

Failure to observe this information can result in injury or equipment damage.

© 2018 Schneider Electric. All rights reserved

# 1 Introduction

This application note describes possibility to control Philips Hue lamps from KNX installation using Wiser for KNX. Philips Hue is RGB LED bulb, which is mainly intended to be controlled from smartphones and tablets.

## 1.1 Added value of this application note

Philips Hue lamps are very state-of-art LED light bulbs, which are becoming very popular. They are mainly controlled by smartphones or special remote controllers. Using Wiser for KNX you can make your installation more complex and unified. You do not need any special application or remote controller, but you do all the setting of your lights directly from Wiser for KNX visualization or using KNX pushbuttons. Next and very important added value is, that you can use your KNX installation and Wiser for KNX logic to create advanced and modern set-up. You can control your Philips Hue lamps to specific light level or you can control your light by movement sensor.

## 1.2 Competencies

This document is intended for readers who have been trained on Wiser for KNX, spaceLYnk products. The integration should not be attempted by someone who is new to the installation of either products. In addition we recommend that you be familiar with:

- The concepts of KNX
- Basic technical knowledge on software technologies such:
  - HTTP, JSON
  - Lua scripting

## 1.3 System prerequisites

Software	Version	Download
Wiser for KNX	2.1 and newer	<a href="http://www.schneider-electric.com">http://www.schneider-electric.com</a>

Table 1: Software versions of used software.

A glossary is available in the appendix chapter of this document. Please refer to it whenever necessary.



## 2 Design

Philips Hue lamps are controlled over ZigBee communication. There is Hue ZigBee Bridge, which translates http requests from LAN network into ZigBee communication (proprietary protocol). Wiser for KNX sends http request on events triggered by KNX devices or Wiser for KNX visualization.

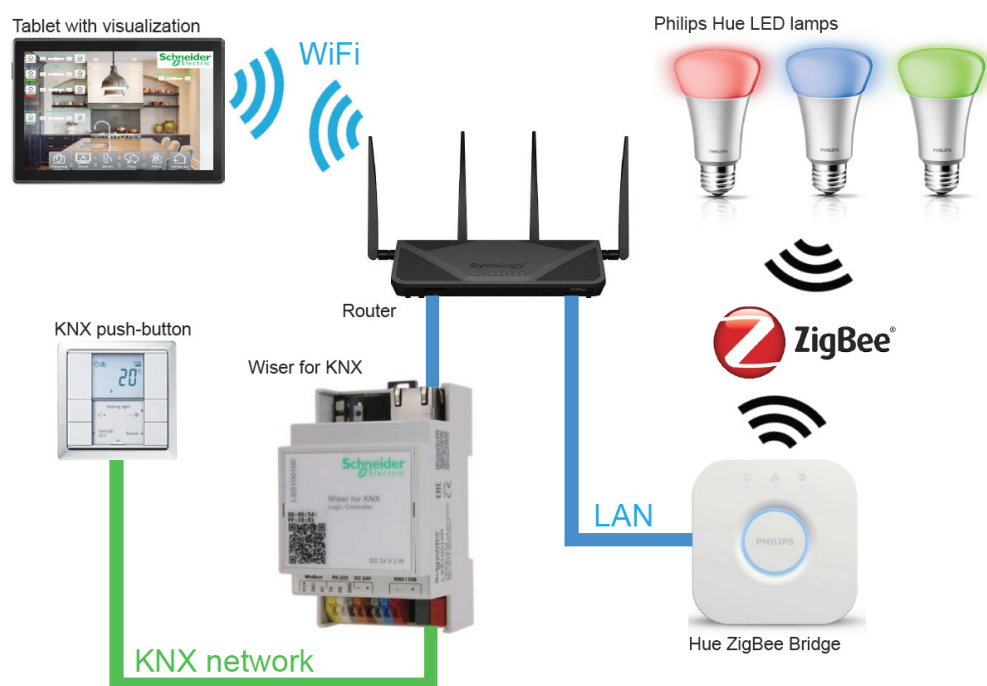


Figure 1: Wiser for KNX - Philips Hue communication scheme.

### 3 Basic user.hue library description

One of classes in user library `user.hue` is class `Bridge`, after its instance is possible to use methods for calling API methods on the bridge.

- **Bridge: get** ([path[, data]])  
 HTTP GET call to the bridge REST API.  
 Makes a HTTP GET call to `path` method of the bridge REST API with given data.
  - `path`: (**string**) Bridge REST API method path. (*optional*)
  - `data`: Optional data to be passed into HTTP request body. (*optional*)
- **Bridge: put** ([path[, data]])  
 HTTP PUT call to the bridge REST API.  
 Makes a HTTP PUT call to `path` method of the bridge REST API with given data.
  - `path`: (**string**) Bridge REST API method path. (*optional*)
  - `data`: Optional data to be passed into HTTP request body. (*optional*)
- **Bridge: post** ([path[, data]])  
 HTTP POST call to the bridge REST API.  
 Makes a HTTP POST call to `path` method of the bridge REST API with given data.
  - `path`: (**string**) Bridge REST API method path. (*optional*)
  - `data`: Optional data to be passed into HTTP request body. (*optional*)
- **Bridge: delete** ([path[, data]])  
 HTTP DELETE call to the bridge REST API.  
 Makes a HTTP DELETE call to `path` method of the bridge REST API with given data.
  - `path`: (**string**) Bridge REST API method path. (*optional*)
  - `data`: Optional data to be passed into HTTP request body. (*optional*)
- **Bridge: set\_rgb** (path, rgb[, brightness[, auto\_onoff=true]])  
 Sets RGB color.  
 Converts a RGB color using **rgb2xy** function and calls a **Bridge: put** method with computed `x`, `y` values and optionally other arguments based on given `brightness` and `auto_onoff` parameters.
  - `path`: (**string**) Bridge REST API method path.
  - `rgb`: (**int** or **string**) RGB color, for more information see RGB argument of **rgb2xy** function.
  - `brightness`: (**int** or **bool**) Brightness to be set. See `bri` argument in [Set light state](#) documentation. When given as boolean=`false` it causes to avoid sending the brightness withing the request at all. (*default*, adjusted brightness from **rgb2xy** call) (*optional*)
  - `auto_onoff`: (**bool**) Whether to send also value for turn the light(s) on/off computed based on given/computed brightness. (*default* true)
- **Bridge: link** ([force=false])  
 Links the **Bridge** instance with the bridge.  
 When the instance is not linked or the `force` parameter is used then it generates a request to the bridge to create a new user.
  - `force`: (**bool**) Whether to make a linking even when the instance is already linked with current (`address`, `user`) configuration. (*default* false)

- **Bridge:unlink** ([force=false])

Unlinks the **Bridge** instance with the bridge.

- **force: (bool)** Whether to make a linking even when the instance is already linked with current (address, user`) configuration. (*default false*)

- **Bridge:is\_linked**()

Linking check. Checks whether the bridge instance with set address and user is really linked to the Hue ZigBee Bridge. Or in other words if the given user is whitelisted on the bridge with specified address.

- **Bridge.register\_user**(address[, devicetype='SE#device MAC'[, use\_https=true]])

Creates new user on the bridge. Registers/creates new user on the hue bridge specified by its IP address or domain name.

- **Bridge.discover** ([timeout=10])

Bridge discovery method. The discovery goes in the steps:

1. Send a SSDP M-SEARCH multicast message for UPnP devices discovery.
2. Wait for the responses.
3. Filter only responses with Hue ZigBee **Bridge** signatures.
4. Create an instance with retrieved bridge information.
5. Put the new Bridge instance into the result array.
6. Return the results.

- **timeout: (int)** Timeout in seconds used in M-SEARCH multicast request. (*default 10*)

## 4 Configuration

This chapter describes Hue ZigBee Bridge and Wiser for KNX network setups.

### 4.1 Hue ZigBee Bridge Setup

**Step 1** Connect Hue ZigBee Bridge to LAN port of router.

**Step 2** Turn on DHCP in your router and choose one following options:

**a. MAC -> IP rule**

- i. Set MAC-IP rule, which ensure, that your Hue ZigBee Bridge will get the same IP address after every request (MAC address of Hue ZigBee Bridge is written on the bottom of the device).
- ii. Keep DHCP functions of your Wi-Fi router turned on.

**b. Use mobile IP to set static IP address of Hue ZigBee Bridge**

- i. Download official Philips Hue app for iOS or Android to your smartphone.
- ii. Connect your smartphone into your home network.
- iii. Search for Hue ZigBee Bridge using the iOS or Android app.
- iv. Turn off DHCP setting of your Hue ZigBee Bridge (it will make its IP address static).
- v. Now you can turn off DHCP functions of your router, if you do not need it for other purposes.

### 4.2 Wiser for KNX and Hue ZigBee Bridge setup

**Step 1** Open Wiser for KNX Configurator in your browser.

**Step 2** Go to *Configurator >> Scripting >> User libraries*.

**Step 3** Click button **Add new library** in left bottom corner on the screen.

**Step 4** Into field *Script name*, write upnp. Uncheck *Auto load library* and click **Save**.

**Step 5** In column *Editor*, click the icon on the row with created upnp library.

**Step 6** Open upnp.lua library attached to this application note and copy the content to this script opened in Step 5.

**Step 7** Click **Save and close**.

**Step 8** Repeat steps 3 - 7 for the second library named hue.

**Step 9** Create e. g. *Event script* and paste there following code:

```
local Bridge = require('user.hue').Bridge
log(Bridge.discover())
```

**Step 10** **Save and Run script**.

**Step 11** In the left bottom corner on the screen click **Logs**. Write down the IP address of discovered bridge(s), it will be needed later. Pay attention information about https. This parameter indicates the bridge's capability to communicate via https protocol. When you see this parameter is set to false, you must explicitly set it to false when instantiating class Bridge.

**Step 12** If you do not have any user whitelisted on the Hue ZigBee Bridge yet or you want to create a new one then press a link button on the bridge and run the following code. Replace address with the IP address of your bridge (the one obtained in previous step).

```
local Hue = require('user.hue')  
local bridge = Hue.Bridge{ address='192.168.0.5' }  
log(bridge:link())
```

**Step 13** You should find a username for newly created user in the **Logs**. In case of error there is an error description instead.

**Step 14** Now you can start to use the Hue REST API. Either create a user library in the *Configurator >> Scripting >> User libraries* (uncheck autoload), name it “mybridge” and paste the following code (replace the address, user and https parameters with actual yours):

```
return require('user.hue').Bridge{  
  address='192.168.0.5',  
  user='gIBY2pQ98kMEDGHQPdK8MmT93HapBEZxPmHy8a72 '  
}
```

and then in the scripts just load the “user.mybridge”

```
local bridge = require('user.mybridge')
```

or keep the following code at the top of your scripts:

```
local Hue = require('user.hue')  
local bridge = Hue.Bridge {  
  address='192.168.0.5',  
  user = 'gIBY2pQ98kMEDGHQPdK8MmT93HapBEZxPmHy8a72 '  
}
```

Now the communication is set and you can start write your commands from Hue library.

## 5 Hue control

### 5.1 Check user existence on the bridge

- To check your user is whitelisted on the bridge run the following code:

```
local Hue = require('user.hue')  
  
local bridge = Hue.Bridge{  
    address='192.168.0.5',  
    user = 'gIBY2pQ98kMEDGHQPdK8MmT93HapBEZxPmHy8a72'  
}  
  
log(('bridge is linked: %s'):format(bridge:is_linked()))
```

*Change IP address and username to one you got during the discovery process.*

If there isn't any user on the bridge, the code returns false to **Logs**.

### 5.2 Delete (unlink) the user from the bridge

```
local Hue = require('user.hue')  
  
local bridge = Hue.Bridge{  
    address='192.168.0.5',  
    user = 'gIBY2pQ98kMEDGHQPdK8MmT93HapBEZxPmHy8a72'  
}  
  
log(bridge:unlink())
```

*Change IP address and username to one you got during the discovery process.*

See the **Logs** for result.

### 5.3 Turn the lamp on with 50% brightness

```
bridge:put('/lights/1/state', {on=true, bri=127})
```

### 5.4 Turn the lamp off

```
bridge:put('/lights/3/state', {on=false})
```

### 5.5 Set lamp color in xy color space

```
bridge:put('/lights/1/state', {xy={0.3972, 0.4564}})
```

### 5.6 Turn the lamp on with brightness, saturation and hue color

```
local data = {on=true, bri=255, hue=46920, sat=255}
bridge:put('/lights/2/state', data)
```

### 5.7 Set color temperature

```
bridge:put('/lights/2/state', {ct=500})
```

### 5.8 Set color using RGB object

```
bridge:set_rgb('/lights/3/state', '11bbff')
```



**Note:** You can use group object of KNX data type 232.600 RGB color. Value of this object can be easily changed in Wiser for KNX visualization using color picker object.

### 5.9 Make all lights blinking for 30s

```
bridge:put('/groups/0/action', {
  on=true,
  sat=255,
  bri=255,
  hue=6144,
  alert='lselect'
})
```



**Note:** Group 0 is implicitly created in Hue ZigBee Bridge. It contains all available lamps. If you want to send command to all lamps, use command `Bridge:put('/groups/0/action', {yourcommand})`.

### 5.10 Put all lights into color loop

```
bridge:put('/groups/0/action', {effect='colorloop'})
```

## 5.11 API dot notation

Instead of using general methods you can use a “dot notation”:

```
bridge:put('/groups/1/action', {bri=50})
bridge:set_rgb('/lights/1/state', 'e2f34a')
bridge:put('/lights/2/state', {on=false})
bridge:put('/lights/3/state', {on=true, bri=190})
```

can be written as:

```
bridge.api.groups[1].action:put{bri=50}
local lights = bridge.api.lights
lights[1].state:set_rgb('e2f34a')
lights[2].state:put{on=false}
-- assignment is syntax sugar for calling the :put() method
lights[3].state = {on=true, bri=190}
```



## 6 Controlling the lights state

There are various properties of Philips Hue lamps configuration, which can be changed using http request. User library `user.hue` contains class `Bridge`. Methods of this class sends http requests. Examples of sending http requests using functions from user library are placed in the previous chapter.

### 6.1 Light state body arguments

You can pass following arguments as part of the data when calling methods to set the lights. For full and up-to-date arguments list visit [https://developers.meethue.com/documentation/lights-api#162\\_body\\_arguments](https://developers.meethue.com/documentation/lights-api#162_body_arguments)

Parameter	Type	Description	Possible values
"on"	bool	Turn the light on/off.	true, false
"bri"	uint8	Brightness of the light. This is a scale from the minimum brightness the light is capable of 1, to the maximum capable brightness, 254.	1..254
"xy"	[float, float]	X and Y coordinates from CIE color space (see picture below). The first entry is the x coordinate and the second entry is the y coordinate. Both x and y are between 0 and 1. Using CIE xy, the colors can be the same on all lamps if the coordinates are within every lamps gamuts (example: xy={0.409,0.5179} is the same color on all lamps). If not, the lamp will calculate it's closest color and use that. The CIE xy color is absolute, independent from the hardware.	[0..1, 0..1]
"hue"	uint16	Hue of the light. This is a wrapping value between 0 and 65535. Note, that hue/sat values are hardware dependent which means that programming two devices with the same value does not guarantee that they will be the same color. Programming 0 and 65535 would mean that the light will resemble the color red, 21845 for green and 43690 for blue.	0..65535
"sat"	uint8	Saturation of the light. 254 is the most saturated (colored) and 0 is the least saturated (white).	0..254
"ct"	uint16	The Mired Color temperature of the light. 2012 connected lights are capable of 153 (6500K) to 500 (2000K).	153..500
"alert"	string	'none' – The light is not performing an alert effect. 'select' – The light is performing one breathe cycle. 'lselect' – The light is performing breathe cycles for 30 seconds or until an alert='none' command is received.	'none', 'lselect', 'select'
"effect"	string	The dynamic effect of the light, can either be "none" or 'colorloop'. If set to colorloop, the light will cycle through all hues using the current brightness and saturation settings.	'none', 'colorloop'
"colormode"	string	Indicates the color mode in which the light is working, this is the last command type it received. Values are "hs" for Hue and Saturation, "xy" for XY and "ct" for Color Temperature. This parameter is only present when the light supports at least one of the values.	'hs', 'xy', 'ct'
"reachable"	bool	Indicates if a light can be reached by the bridge.	true, false

Table 2: Parameters for calling API methods.

In order to better understand the setting of Philips Hue lamps it is needed to explain the color space interpretation at first. The picture below shows the CIE color space and the green triangle outlines the colors, which can be produced by Philips Hue lamps. You can see that some highly saturated colors and almost all green tones cannot be reached using Philips Hue lamps.

There are 3 different ways how to determine the requested color of the light:

## 6.2 XY coordinates in CIE color space

Select the color from CIE color space using XY coordinates. CIE color space is displayed in Figure 2. If you select color outside the green triangle, closest color inside the triangle will be chosen.

Color is fully specified by two parameters: X and Y.

## 6.3 Hue color + saturation

The hue value is a wrapping value between 0 and 65535. Both 0 and 65535 are red, 25500 is green and 46920 is blue. See Figure 3. Saturation of the color can be set from 0..255. Full saturation basically goes around the green triangle in Figure 2. As the saturation value is decreasing, colors are going to be less intense and more white. In other words, decreasing the saturation value moves the color from the triangle edge towards the white point inside the triangle.

Color is fully specified by two parameters: Hue and saturation.

## 6.4 White color temperature

Black curve in the middle of Figure 2 follows white colors from the cold white to warm white. Philips Hue lamps can produce white light between 2000K (warm) and 6500K (cold). The temperature of the white color can be specified by “ct” parameter (color temperature), which is scaled using scale named “reciprocal megakelvin” or “mirek”. Using this scale, the warmest white 2000K corresponds to 500mirek (ct=500) and the coldest white 6500K to 153mirek (ct=153).

Color is fully specified by one parameter: color temperature.

## 6.5 RGB color setting

Wiser for KNX, as KNX oriented device supports KNX DPT 232.600 for RGB color. It can be useful and easier to specify color using RGB specification rather than any of three options listed above. For this case there has been created class method `Bridge:set_rgb(path, rgb[, brightness[, auto_onoff=true]]`), which is part of user library `user.hue`. This function transform RGB value into XY coordinates and send it to a specified Hue lamp.

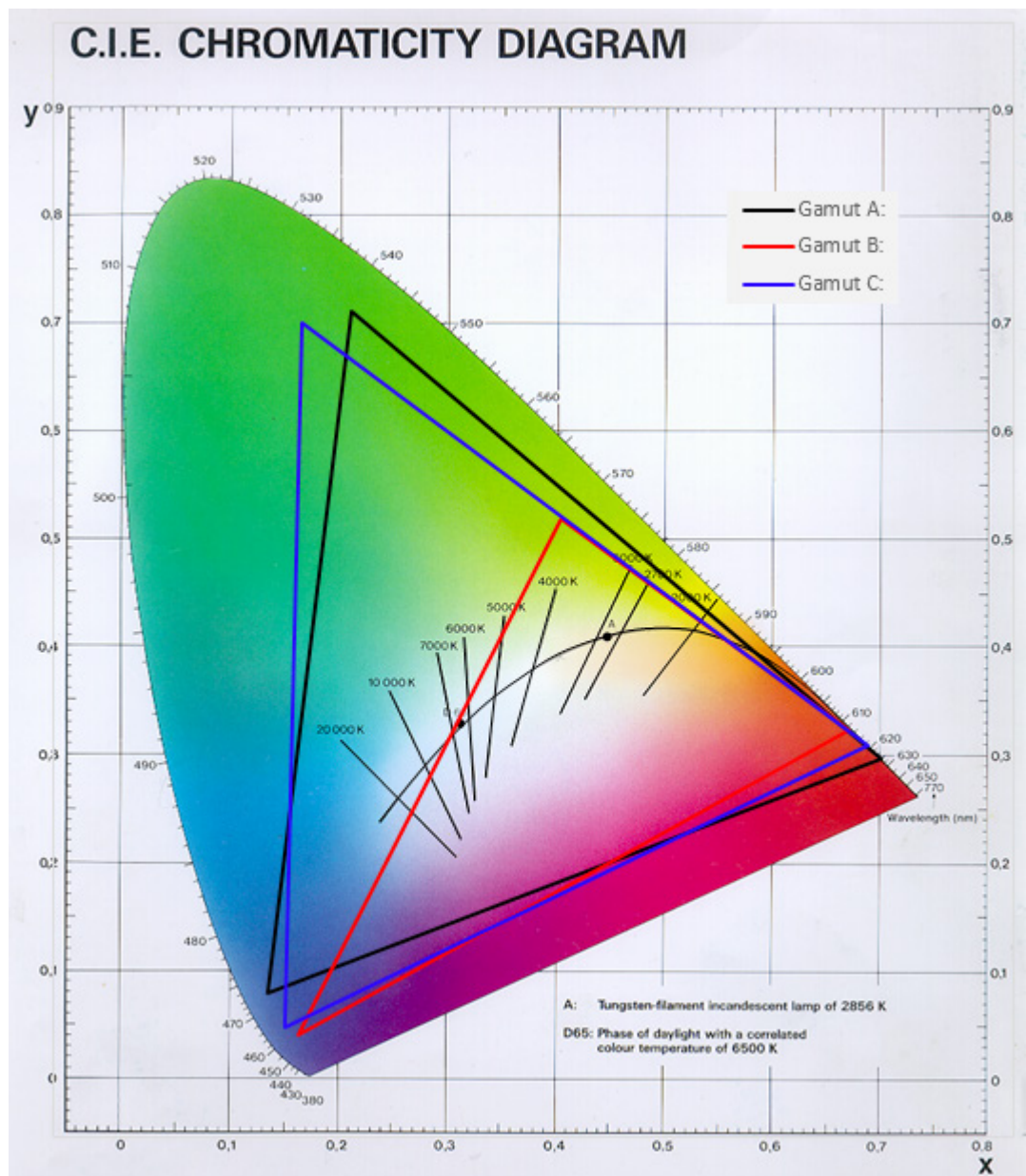


Figure 2: CIE color space of Philips Hue lamps.

Hue	Final-x	Final-y
0	0.3972	0.4564
12750	0.5425	0.4196
25500	0.41	0.51721
46920	0.1691	0.0441
56100	0.4149	0.1776
65280	0.6679	0.3181

Figure 3: Common Hue values and corresponding XY values.

Pictures above and detail description can be found here: <https://developers.meethue.com/documentation/core-concepts>

## 7 Sample project

This application note contains ready to use example of Philips Hue integration with Wiser for KNX. To use the example follow the steps below:

- Step 1** Go to *Configurator >> Utilities >> Restore*.
- Step 2** Choose file to restore: AN2\_002\_Philips\_Hue\_W4K\_backup.tar.gz and click **Save**.
- Step 3** Go to *Configurator >> Scripting >> Event-based*.
- Step 4** Open *bridge\_discovery* script in script editor.
- Step 5** Click button **Run script** in the right bottom corner.
- Step 6** Click button **Logs** in the left bottom corner and write down IP address of the bridge. Pay attention the parameter https (described in Step 11, chapter 4.2). Close this script.
- Step 7** Open *bridge\_link* script in script editor.
- Step 8** Press link button on the top of your Hue ZigBee Bridge.
- Step 9** Run *bridge\_link* script by click the button **Run script** in the right bottom corner.
- Step 10** Click on **Logs** and write down the username. Close the script.
- Step 11** Go to *User libraries* and open script *user.mybridge* in script editor.
- Step 12** Change the IP address and user to the one you received in previous steps and save the library.
- Step 13** Example is prepared for use to control Philips Hue lamps.



Figure 4: Visualization of Wiser for KNX usage example.

## 8 Conclusion

This application note provides step-by-step description how to pair your Wiser for KNX with Hue ZigBee Bridge and your Philips Hue lamps. Very important part is user library `user.hue`, which helps to easy control Philips Hue lamps from Wiser for KNX scripts. The main options of Philips Hue lamps are described above and all of them are used in examples. If you are looking for more detailed description of Hue API, please refer to the Hue API documentation website, which is listed in reference of this application note.

## 9 Appendix

In this chapter you can find useful links for tools, firmware and other things which can help to successfully go through this application note.

### 9.1 Glossary

The following table describes the acronyms and defines the specific terms used in this document.

Abbreviation	Description
DHCP	Dynamic Host Configuration Protocol
HTTP	HyperText Transfer Protocol
JSON	JavaScript Object Notation
RGB	Red Green Blue (color representation)
Lua	Programming language
REST API	Representational State Transfer Application Programming Interface
CIE	Color space using XY coordinates
UPnP	Universal Plug and Play
SSDP	Simple Service Discovery Protocol

Table 3: Specific terms.

### 9.2 Reference

Document title	Reference
Wiser for KNX User Guide	<a href="http://download.schneider-electric.com/files?p_enDocType=User+guide&amp;p_File_Name=AR1740_Edl_User_Wiser_for_KNX_EN.pdf&amp;p_Doc_Ref=AR1740_Edl_EN">http://download.schneider-electric.com/files?p_enDocType=User+guide&amp;p_File_Name=AR1740_Edl_User_Wiser_for_KNX_EN.pdf&amp;p_Doc_Ref=AR1740_Edl_EN</a>
Hue API documentation	<a href="http://www.developers.meethue.com/philips-hue-api">http://www.developers.meethue.com/philips-hue-api</a>

Table 4: Reference.



## Schneider Electric

35 rue Joseph Monier

92500 Rueil Malmaison – France

Phone: +33 (0) 1 41 29 70 00

[www.schneider-electric.com](http://www.schneider-electric.com)

As standards, specifications, and designs change from time to time, please ask for confirmation of the information given in this publication.

© 2014-2018 Schneider Electric. All rights reserved.

LSS100100

Rev. 1



This document has been  
printed on recycled paper.