Installing the Gateway

Contents

[Setting up before installing OpenHAB 2](#_Toc471476622)

[Installing OpenHAB 3](#_Toc471476623)

[Create the install folder 3](#_Toc471476624)

[Get the source code from the repository 3](#_Toc471476625)

[Compile the source code 3](#_Toc471476626)

[Extract the archive 3](#_Toc471476627)

[First run 3](#_Toc471476628)

[Initialization 3](#_Toc471476629)

[Sitemap 3](#_Toc471476630)

[Create the new sitemap 3](#_Toc471476631)

[Set as default sitemap 4](#_Toc471476632)

[Hardware 4](#_Toc471476633)

[Installing extensions 4](#_Toc471476634)

[MQTT as persistence 4](#_Toc471476635)

[Install mosquito broker 4](#_Toc471476636)

[Install the MQTT persistence service 4](#_Toc471476637)

[Configure connection to the MQTT broker 4](#_Toc471476638)

[Configure MQTT persistence 5](#_Toc471476639)

[Setup new persistence strategy 5](#_Toc471476640)

[MQTT as eventbus binding 5](#_Toc471476641)

[Installing the MQTT binding 6](#_Toc471476642)

[Configure MQTT eventbus 6](#_Toc471476643)

[Installing protocols 6](#_Toc471476644)

[KNX 6](#_Toc471476645)

[Install the KNX binding 6](#_Toc471476646)

[Configure connection to the KNX/IP gateway 7](#_Toc471476647)

[Configure KNX items 7](#_Toc471476648)

[Add items to the sitemap 7](#_Toc471476649)

[EnOcean 8](#_Toc471476650)

[Install binding dependencies 8](#_Toc471476651)

[Install binding 9](#_Toc471476652)

[Configure the binding 10](#_Toc471476653)

[Configure EnOcean items 10](#_Toc471476654)

[Add items to the sitemap 11](#_Toc471476655)

[Z-Wave 12](#_Toc471476656)

[Install the Z-Wave binding 12](#_Toc471476657)

[Add the Z-Wave controller 12](#_Toc471476658)

[Add the Z-Wave sensors 14](#_Toc471476659)

[Add the sensors to the sitemap 14](#_Toc471476660)

[Nest 15](#_Toc471476661)

[Install the Nest binding 15](#_Toc471476662)

[Get required identification data 15](#_Toc471476663)

[Configure the binding 15](#_Toc471476664)

[Configure the associated items 15](#_Toc471476665)

[Troubleshooting 16](#_Toc471476666)

[Logging 16](#_Toc471476667)

[Display issues 16](#_Toc471476668)

[KNX binding 16](#_Toc471476669)

[Timeout 16](#_Toc471476670)

[Item definition not refreshing 16](#_Toc471476671)

[EnOcean binding 16](#_Toc471476672)

[Port not found 16](#_Toc471476673)

[Z-Wave binding 16](#_Toc471476674)

[Devices showing as Unknown Device 16](#_Toc471476675)

[Badly detected devices 16](#_Toc471476676)

[Nodes added, but no items bound 16](#_Toc471476677)

[PST-02A door status not updating 17](#_Toc471476678)

[Aeotec Multisensor 6 not sending events 17](#_Toc471476679)

[Aeotec Key Fob Gen5 17](#_Toc471476680)

# Setting up before installing OpenHAB

For Windows, install Chocolatey as it will be used to install dependencies. Windows commands assume using the default command prompt (cmd.exe).

Tutorial here: <https://chocolatey.org/install>

Java based tool so should work with any OS with a Java runtime. Successfully tested with Java JDK 8u60.

Install it on Windows by running:

choco install jdk8 --version 8.0.60

setx JAVA\_HOME “C:\Program Files\Java\jdk1.8.0\_60”

refreshenv

To get OH from source, maven has to be installed along with git. 7zip extracts the compiled archive. Notepad++ and nano are used to edit configuration files. Chromium is needed on Windows because IE and Edge cannot display OpenHAB properly. Gradle is used to build dependencies.

On Windows:

choco install git maven 7zip.commandline notepadplusplus nano chromium gradle

refreshenv

On Debian:

apt-get install git maven p7zip-full chromium-browser gradle

# Installing OpenHAB

### Create the install folder

Windows:

mkdir C:\openhab

cd C:\openhab

Linux:

mkdir ~/openhab

cd ~/openhab

### Get the source code from the repository

git clone <https://github.com/openhab/openhab-distro.git>

cd openhab-distro

### Compile the source code

mvn install

The process can take 30 minutes to complete, depending on how fast your computer and internet connection are. The generated archive will be located in

/openhab/openhab-distro/distributions/openhab-{offline:online}/target/

### Extract the archive

For the rest of the setup we are going to use the offline distribution, which includes a local copy of the bindings.

Windows:

cd distributions\openhab-offline\target\

7z x openhab-offline-2.0.0-SNAPSHOT.zip -oC:\openhab\openhab-snapshot

Linux:

cd distributions/openhab-offline/target/

7z x openhab-offline-2.0.0-SNAPSHOT.zip -o$HOME/openhab/openhab-snapshot

### First run

Do a first run of OpenHAB to complete the initial configuration.

Windows:

cd C:\openhab\openhab-snapshot

start.bat

Linux:

cd ~/openhab/openhab-snapshot

./start.sh

The extensions should initialize in a few minutes. To exit OpenHAB, issue Ctrl-D to its shell.

# Initialization

## Sitemap

### Create the new sitemap

Windows:

cd C:\openhab\openhab-snapshot\conf\sitemaps

start notepad++ demo.sitemap

Linux:

cd ~/openhab/openhab-snapshot/conf/sitemaps

nano demo.sitemap

Fill the file as follows:

sitemap demo label="Demo"

{

}

Save and exit (in Nano: Ctrl-X, Y, Enter).

### Set as default sitemap

To be usable in the UI, it has to be set as current sitemap using the OpenHAB web interface:

<http://localhost:8080/ui/index.html#/configuration/services>

Click the Configure button in Basic UI and set the default sitemap field to *demo*.

## Hardware

Plug the EnOcean USB300 stick into the computer, then plug the Z-Wave controller stick into the computer.

# Installing extensions

## MQTT as persistence

Install MQTT persistence allows broadcasting of all item state changes with OpenHAB generated timestamps. This is best used to log data from OpenHAB.

### Install mosquito broker

If required, download and run Mosquitto to have an MQTT broker.

Windows:

Follow this tutorial:

<https://sivatechworld.wordpress.com/2015/06/11/step-by-step-installing-and-configuring-mosquitto-with-windows-7/>

Debian:

apt-get install mosquitto

### Install the MQTT persistence service

With OpenHAB running, open the web interface:

<http://localhost:8080/ui/index.html#/extensions>

Select the persistence tab, find MQTT and install it:



### Configure connection to the MQTT broker

Once installed, you have to edit the configuration file to setup connection to the MQTT broker.

Windows:

cd C:\openhab\openhab-snapshot\conf\services

start notepad++ mqtt.cfg

Linux:

cd ~/openhab/openhab-snapshot/conf/services

nano mqtt.cfg

Replace line 7 of the configuration:

#<broker>.url=tcp://<host>:1883

By

mosquitto.url=tcp://localhost:1883

The other parameters are optional. Save and close the file.

### Configure MQTT persistence

Then open mqtt-persistence.cfg:

Windows:

cd C:\openhab\openhab-snapshot\conf\services

start notepad++ mqtt-persistence.cfg

Linux:

cd ~/openhab/openhab-snapshot/conf/services

nano mqtt-persistence.cfg

And set the contents as follows:

# Name of the broker as defined in mqtt.cfg

broker=mosquitto

# The MQTT topic to which the persistence messages should be sent.

topic=openhab/%1$s

# A string representing the persistence message content.

message=%4$s,%1$s,%3$s

### Setup new persistence strategy

Create new persistence strategy:

Windows:

cd C:\openhab\openhab-snapshot\conf\persistence

start notepad++ mqtt.persist

Linux:

cd ~/openhab/openhab-snapshot/conf/persistence/

nano mqtt.persist

File contents:

// MQTT persistence strategy

Strategies {

 default = everyChange

}

Items {

 \* : strategy = everyChange

}

Save the file.

OpenHAB should now send all changes to topic openhab on the local mosquitto MQTT broker.

## MQTT as eventbus binding

Installing MQTT as eventbus binding allows to receive updates from OpenHAB (albeit without timestamps) but most importantly to send commands to OpenHAB and set item states using an MQTT publisher.

### Installing the MQTT binding

 With OpenHAB running, open the web interface:

<http://localhost:8080/ui/index.html#/extensions>

Select the Bindings tabs and find the MQTT binding:



Then click the Install button.

### Configure MQTT eventbus

Windows:

cd C:\openhab\openhab-snapshot\conf\services

start notepad++ mqtt-eventbus.cfg

Linux:

cd ~/openhab/openhab-snapshot/conf/services

nano mqtt-eventbus.cfg

Fill the content of the file as follows:

broker=mosquitto

statePublishTopic=openhab/${item}/state

commandPublishTopic=openhab/${item}/commands

stateSubscribeTopic=openhab/${item}/statesub

commandSubscribeTopic=openhab/${item}/commandssub

Save the file. OpenHAB should send data to topics ending in state and commands, and receive updates from statesub and commandssub. It is recommended to send commands using the commandssub topic as states are not properly propagated to physical items.

# Installing protocols

## KNX

### Install the KNX binding

With OpenHAB running, open the web interface:

<http://localhost:8080/ui/index.html#/extensions>

Select the Bindings tabs and find the KNX binding:



Then click Install.

### Configure connection to the KNX/IP gateway

Then open the KNX configuration file:

Windows:

cd C:\openhab\openhab-snapshot\conf\services

start notepad++ knx.cfg

Linux:

cd ~/openhab/openhab-snapshot/conf/services

nano knx.cfg

Change the values of the following settings, assuming the KNX/IP gateway IP is 192.168.1.9:

ip=192.168.1.9

busaddr=1.1.1

ignorelocalevents=true

type=TUNNEL

localIp=<ip openhab>

To retrieve the computer’s IP, issue the following commands:

Windows:

ipconfig

Linux:

ifconfig

###

### Configure KNX items

Create KNX item files:

Windows:

cd C:\openhab\openhab-snapshot\conf\items

start notepad++ 27d-knx.items

Linux:

cd ~/openhab/openhab-snapshot/conf/items

nano 27d-knx.items

Copy this content into the item file:

Switch sample\_item "KNX Item" { knx="0/0/1+0/1/1" }

Save the file. OpenHAB should reload it automatically.

### Add items to the sitemap

Open the sitemap:

Windows:

cd C:\openhab\openhab-snapshot\conf\sitemaps

start notepad++ 27d.sitemap

Linux:

cd ~/openhab/openhab-snapshot/conf/sitemaps

nano 27d.sitemap

Add this between the two brackets:

Frame label="Communication KNX"{

 Switch item=sample\_item

}

Save the file and load the basic UI:

<http://localhost:8080/basicui/app>

OpenHAB should load the sitemap with all the items:



## EnOcean

### Install binding dependencies

Before installing the dependencies, make sure OpenHAB is running and follow the instructions as the load order of the dependencies matters.

#### NRJavaSerial

Setup the build folder:

Windows:

cd C:\openhab\

mkdir deps

git clone <https://github.com/NeuronRobotics/nrjavaserial.git>

Linux:

cd ~/openhab/

mkdir deps

git clone <https://github.com/NeuronRobotics/nrjavaserial.git>

Build the project:

cd nrjavaserial

gradle install

Copy the dependency inside OpenHAB:

Windows:

copy build\libs\nrjavaserial-3.12.0.jar C:\openhab\openhab-snapshot\addons

Linux:

cp build/libs/nrjavaserial-3.12.0.jar ~/openhab/openhab-snapshot/addons

Make sure NRJavaSerial has been successfully loaded by switching to the OpenHAB prompt and running:

bundle:list

NRJavaSerial should appear in the list:

192 | Active | 80 | 3.12.0 | nrjavaserial

#### Aleoncean

Setup the build folder:

Windows:

cd C:\openhab\deps

git clone <https://github.com/aleon-GmbH/aleoncean.git>

Linux:

cd ~/openhab/deps

git clone <https://github.com/aleon-GmbH/aleoncean.git>

Build the project:

cd aleoncean

mvn install

Copy the dependency inside OpenHAB:

Windows:

copy target\aleoncean-1.10-SNAPSHOT.jar C:\openhab\openhab-snapshot\addons

Linux:

cp target/aleoncean-1.10-SNAPSHOT.jar ~/openhab/openhab-snapshot/addons

Make sure Aleoncean has been successfully loaded by switching to the OpenHAB prompt and running:

bundle:list

Aleoncean should appear in the list:

193 | Active | 80 | 1.10.0.SNAPSHOT | aleoncean

### Install binding

Before installing the binding make sure both dependencies are up and running.

Setup the build folder:

Windows:

cd C:\openhab\deps

git clone <https://github.com/aleon-GmbH/aleoncean-openhab-binding.git>

Linux:

cd ~/openhab/deps

git clone <https://github.com/aleon-GmbH/aleoncean-openhab-binding.git>

Edit the Maven configuration file to use our version of aleoncean:

Windows:

cd aleoncean-openhab-binding

start notepad++ pom.xml

Linux:

cd aleoncean-openhab-binding

nano pom.xml

Scroll to property aleoncean.version on line 60 and set it to 1.10-SNAPSHOT:

 <aleoncean.version>1.10-SNAPSHOT</aleoncean.version>

Save and exit.

Build the project:

cd aleoncean-openhab-binding

mvn install

Copy the dependency inside OpenHAB:

Windows:

copy target\org.openhab.binding.aleoncean-1.7.5-SNAPSHOT.jar C:\openhab\openhab-snapshot\addons

Linux:

cp target/org.openhab.binding.aleoncean-1.7.5-SNAPSHOT.jar ~/openhab/openhab-snapshot/addons

Make sure it has been successfully loaded by switching to the OpenHAB prompt and running:

bundle:list

The binding should appear in the list:

194 | Active | 80 | 1.7.5.SNAPSHOT | openHAB aleoncean Binding

### Configure the binding

Create Aleoncean binding configuration:

Windows:

cd C:\openhab\openhab-snapshot\conf\services

start notepad++ aleoncean.cfg

Linux:

cd ~/openhab/openhab-snapshot/conf/services

nano aleoncean.cfg

Add the following content inside the file, changing the COM port depending on where the USB300 stick is:

# EnOcean USB adapter serial port

port=COM3

Save the file, OpenHAB should try to connect to the USB stick.

### Configure EnOcean items

Create EnOcean item files:

Windows:

cd C:\openhab\openhab-snapshot\conf\items

start notepad++ 27d-enocean.items

Linux:

cd ~/openhab/openhab-snapshot/conf/items

nano 27d-enocean.items

Complete the contents of the documents as follows:

Number ROCKER\_TEST\_A\_N "R (A) [%d]" {aleoncean="REMOTEID=00:25:7f:2d,TYPE=RD\_F6-02-01,PARAMETER=BUTTON\_DIM\_A"}

Switch ROCKER\_TEST\_A\_1 "R (A) up: pressed / released" {aleoncean="REMOTEID=00:25:7f:2d,TYPE=RD\_F6-02-01,PARAMETER=BUTTON\_DIM\_A,CONVPARAM=UpPressedReleased"}

Switch ROCKER\_TEST\_A\_2 "R (A) down: pressed / released" {aleoncean="REMOTEID=00:25:7f:2d,TYPE=RD\_F6-02-01,PARAMETER=BUTTON\_DIM\_A,CONVPARAM=DownPressedReleased"}

Switch ROCKER\_TEST\_A\_3 "R (A) pressed: up / down" {aleoncean="REMOTEID=00:25:7f:2d,TYPE=RD\_F6-02-01,PARAMETER=BUTTON\_DIM\_A,CONVPARAM=PressedUpDown"}

Switch ROCKER\_TEST\_A\_4 "R (A) released: up / down" {aleoncean="REMOTEID=00:25:7f:2d,TYPE=RD\_F6-02-01,PARAMETER=BUTTON\_DIM\_A,CONVPARAM=ReleasedUpDown"}

Number ROCKER\_TEST\_B\_N "R (B) [%d]" {aleoncean="REMOTEID=00:25:7f:2d,TYPE=RD\_F6-02-01,PARAMETER=BUTTON\_DIM\_B"}

Switch ROCKER\_TEST\_B\_1 "R (B) up: pressed / released" {aleoncean="REMOTEID=00:25:7f:2d,TYPE=RD\_F6-02-01,PARAMETER=BUTTON\_DIM\_B,CONVPARAM=UpPressedReleased"}

Switch ROCKER\_TEST\_B\_2 "R (B) down: pressed / released" {aleoncean="REMOTEID=00:25:7f:2d,TYPE=RD\_F6-02-01,PARAMETER=BUTTON\_DIM\_B,CONVPARAM=DownPressedReleased"}

Switch ROCKER\_TEST\_B\_3 "R (B) pressed: up / down" {aleoncean="REMOTEID=00:25:7f:2d,TYPE=RD\_F6-02-01,PARAMETER=BUTTON\_DIM\_B,CONVPARAM=PressedUpDown"}

Switch ROCKER\_TEST\_B\_4 "R (B) released: up / down" {aleoncean="REMOTEID=00:25:7f:2d,TYPE=RD\_F6-02-01,PARAMETER=BUTTON\_DIM\_B,CONVPARAM=ReleasedUpDown"}

Switch EnOceanSmartPlug "prise [%d]" {aleoncean="LOCALID=00:85:E1:DB,REMOTEID=01:92:A4:9E,TYPE=RD\_D2-01-08,PARAMETER=SWITCH"}

Number ubiwizz\_switch\_n "ubiwizz switch [%d]" {aleoncean="REMOTEID=00:29:ca:ed,TYPE=RD\_F6-02-01,PARAMETER=BUTTON\_DIM\_B"}

Switch ubiwizz\_switch\_b\_left "ubiwizz switch left pressed: up / down" {aleoncean="REMOTEID=00:29:ca:ed,TYPE=RD\_F6-02-01,PARAMETER=BUTTON\_DIM\_A,CONVPARAM=PressedUpDown"}

Switch ubiwizz\_switch\_b\_right "ubiwizz switch right pressed: up / down" {aleoncean="REMOTEID=00:29:ca:ed,TYPE=RD\_F6-02-01,PARAMETER=BUTTON\_DIM\_B,CONVPARAM=PressedUpDown"}

Number blank\_switch\_n "blank switch [%d]" {aleoncean="REMOTEID=00:2a:dc:99,TYPE=RD\_F6-02-01,PARAMETER=BUTTON\_DIM\_B"}

Switch blank\_switch\_b\_left "blank switch pressed: up / down" {aleoncean="REMOTEID=00:2a:dc:99,TYPE=RD\_F6-02-01,PARAMETER=BUTTON\_DIM\_B,CONVPARAM=PressedUpDown"}

Save the file. OpenHAB should reload it automatically.

### Add items to the sitemap

Open the sitemap:

Windows:

cd C:\openhab\openhab-snapshot\conf\sitemaps

start notepad++ 27d.sitemap

Linux:

cd ~/openhab/openhab-snapshot/conf/sitemaps

nano 27d.sitemap

Add the following content inside the file, after the KNX content but before the closing bracket:

 Frame label="EnOcean"{

 Switch item=ROCKER\_TEST\_A\_1

 Switch item=ROCKER\_TEST\_A\_2

 Switch item=ROCKER\_TEST\_B\_3

 Switch item=ROCKER\_TEST\_B\_4

 Switch item=EnOceanSmartPlug

 Switch item=blank\_switch\_b\_left

 Switch item=ubiwizz\_switch\_b\_left

 Switch item=ubiwizz\_switch\_b\_right

}

Save the file and load the basic UI:

<http://localhost:8080/basicui/app>

OpenHAB should load the sitemap with the new items at the bottom:



## Z-Wave

### Install the Z-Wave binding

With OpenHAB running, open the web interface:

<http://localhost:8080/ui/index.html#/extensions>

Select the Bindings tabs and find the Z-Wave binding:



Click on Install to install it.

### Add the Z-Wave controller

Open the things configuration screen, either by using the menu on the left after installing the binding or using this link:

<http://localhost:8080/ui/index.html#/configuration/things>



Then click the + button to add a new thing



Select the Z-Wave binding



Select the Z-Wave Controller



Set the Thing ID to 15972cc4660 to ease the setup of the sitemap. Set the COM port to the port used by the USB stick then press the blue checkbox button. The Z-Wave controller should appear in the Things list as follows:



### Add the Z-Wave sensors

From the Things configuration panel, click on the + button and select Z-Wave like before. A list of detected sensors should appear on your screen:



Click each blue checkbox button then validate the modal to add the sensors. Some devices may not be detected properly, or have no bound items (this can be checked by opening the device’s configuration), in which case please refer to the Troubleshooting section at the end of this document for advice on how to proceed for these devices.

### Add the sensors to the sitemap

Open the sitemap:

Windows:

cd C:\openhab\openhab-snapshot\conf\sitemaps\

start notepad++ 27d.sitemap

Linux:

cd ~/openhab/openhab-snapshot/conf/sitemaps

nano 27d.sitemap

Add the content after the EnOcean binding but before the closing bracket:

 Frame label="Z-Wave / PST02A Multi-Sensor (Node 2)"{

 Switch item=zwave\_device\_15972cc4660\_node2\_sensor\_door

 Text item=zwave\_device\_15972cc4660\_node2\_sensor\_luminance

 Text item=zwave\_device\_15972cc4660\_node2\_sensor\_temperature

 }

 Frame label="Z-Wave / The Button (Node 3)"{

 Text item=zwave\_device\_15972cc4660\_node3\_scene\_number

 }

 Frame label="Z-Wave / Prise pilotable (Node 4)"{

 Switch item=zwave\_device\_15972cc4660\_node4\_switch\_binary

 Text item=zwave\_device\_15972cc4660\_node4\_meter\_kwh

 Text item=zwave\_device\_15972cc4660\_node4\_meter\_watts

 }

 Frame label="Z-Wave / Motion sensor (Node 5)"{

 Switch item=zwave\_device\_15972cc4660\_node5\_sensor\_binary

 Text item=zwave\_device\_15972cc4660\_node5\_sensor\_temperature

 Text item=zwave\_device\_15972cc4660\_node5\_sensor\_luminance

 }

The devices should appear in the sitemap:

<http://localhost:8080/basicui/app>

## Nest

### Install the Nest binding

With OpenHAB running, open the web interface:

<http://localhost:8080/ui/index.html#/extensions>

Select the Bindings tabs and find the Nest binding, then install it.

### Get required identification data

Follow the instructions on the binding documentation to get the identification data, consisting of a product ID, a secret and a PIN code.

You may also use the 27Delvalle product that already exists. In that case, take note of the following data:

Product ID: ca059987-c8ec-4f7f-a998-9eb57e7c9e17

Product Secret: UvQqGbhVMTAi37ajFEgXhCBl8

PIN code: RECCAZQW

### Configure the binding

If required, create the configuration file:

conf/services/nest.cfg

Inside this file, fill out the required information:

# the Nest Client ID needed to use the API, must be supplied

client\_id=ca059987-c8ec-4f7f-a998-9eb57e7c9e17

# the Nest Client Secret needed to use the API, must be supplied

client\_secret=UvQqGbhVMTAi37ajFEgXhCBl8

# the PIN code that Nest presented when you authorized the above client, must be supplied

pin\_code=RECCAZQW

Save the file.

### Configure the associated items

For the test camera, you can use the following items to be put inside a new item file

conf/items/nest.items

Items:

String NestCamDeviceId "CamDeviceId [%s]" {nest="<[cameras(Living Room).device\_id]"}

String NestCamSWVersion "CamSWVersion [%s]" {nest="<[cameras(Living Room).software\_version]"}

String NestCamName "CamName [%s]" {nest="<[cameras(Living Room).name]"}

String NestCamNameLong "CamNameLong [%s]" {nest="<[cameras(Living Room).name\_long]"}

Switch NestCamIsOnline "CamIsOnline [%s]" {nest="<[cameras(Living Room).is\_online]"}

Switch NestCamIsStreaming "CamIsStreaming [%s]" {nest="=[cameras(Living Room).is\_streaming]"}

Switch NestCamIsAudioInputEnabled "CamIsAudioInputEnabled [%s]" {nest="<[cameras(Living Room).is\_audio\_input\_enabled]"}

DateTime NestCamLastIsOnlineChange "CamLastIsOnlineChange [%1$tm/%1$td %1$tH:%1$tM]" {nest="<[cameras(Living Room).last\_is\_online\_change]"}

Switch NestCamIsVideoHistoryEnabled "CamIsVideoHistoryEnable [%s]" {nest="<[cameras(Living Room).is\_video\_history\_enabled]"}

String NestCamWebUrl "CamWebUrl [%s]" {nest="<[cameras(Living Room).web\_url]"}

String NestCamAppUrl "CamAppUrl [%s]" {nest="<[cameras(Living Room).app\_url]"}

Switch NestCamIsPublicShareEnabled "IsPublicShareEnabled [%s]" {nest="<[cameras(Living Room).is\_public\_share\_enabled]"} // as of 1.9

String NestCamPublicShareUrl "PublicShareUrl [%s]" {nest="<[cameras(Living Room).public\_share\_url]"} // as of 1.9

String NestCamSnapshotUrl "SnapshotUrl [%s]" {nest="<[cameras(Living Room).snapshot\_url]"} // as of 1.9

/\* All last\_event.\* binding config strings require a Nest Aware with Video History subscription \*/

Switch NestCamLastEventHasSound "CamLastEventHasSound [%s]" {nest="<[cameras(Living Room).last\_event.has\_sound]"}

Switch NestCamLastEventHasMotion "CamLastEventHasMotion [%s]" {nest="<[cameras(Living Room).last\_event.has\_motion]"}

DateTime NestCamLastEventStartTime "CamLastEventStartTime [%1$tm/%1$td %1$tH:%1$tM]" {nest="<[cameras(Living Room).last\_event.start\_time]"}

DateTime NestCamLastEventEndTime "CamLastEventEndTime [%1$tm/%1$td %1$tH:%1$tM]" {nest="<[cameras(Living Room).last\_event.end\_time]"}

DateTime NestCamLastEventUrlsExpireTime "CamLastEventUrlsExpireTime [%1$tm/%1$td %1$tH:%1$tM]" {nest="<[cameras(Living Room).last\_event.urls\_expire\_time]"}

String NestCamLastEventWebUrl "CamLastEventWebUrl [%s]" {nest="<[cameras(Living Room).last\_event.web\_url]"}

String NestCamLastEventAppUrl "CamLastEventAppUrl [%s]" {nest="<[cameras(Living Room).last\_event.app\_url]"}

String NestCamLastEventImageUrl "CamLastEventImageUrl [%s]" {nest="<[cameras(Living Room).last\_event.image\_url]"}

String NestCamLastEventAnimatedImageUrl "CamLastEventAnimatedImageUrl [%s]" {nest="<[cameras(Living Room).last\_event.animated\_image\_url]"}

# Troubleshooting

## Logging

At the OpenHAB command shell, execute

log:tail

This will display the OpenHAB log file.

## Display issues

If accented characters are not working properly, make sure the items file is saved as UTF8 without BOM. This can be checked using Notepad++’s Encoding menu.

## KNX binding

### Timeout

In some cases, the KNX binding can timeout on initialization (as shown in the log) if the localIp parameter is not set correctly. On any failed connection attempt the KNX/IP Gateway may take a few minutes before responding.

### Item definition not refreshing

If the item isn’t updated with a different KNX configuration in the item file, restart OpenHAB.

## EnOcean binding

### Port not found

In some cases, the port may not be accessible because of access rights issues on both OS. Make sure the process can access the port. You may also need to add a launch parameter to whitelist the port if it’s using a non-standard name in Linux.

## Z-Wave binding

### Devices showing as Unknown Device

Either wait a few minutes with the device close the controller until detection is done or delete the device and re-add it after interacting with the device to make sure its type is known.

### Badly detected devices

The default Paper UI can have issues successfully detecting and setting up some Z-Wave devices. These devices may show up as Unknown Device despite several attempts at waking them up, or the configuration page may not allow certain configurations. If this happens, an alternative UI named HABmin can help.

Install HABmin from the User Interfaces tab inside the extension install page:

<http://localhost:8080/ui/index.html#/extensions>

Then navigate to it:

<http://localhost:8080/habmin/index.html#/things>

Once installed, the Things page (linked right above) should display at least the Z-Wave Serial Controller and the Nodes installed from Paper UI. To trigger a network wide inclusion and add new devices to the Z-Wave network, click the scan button at the top of the UI.

To set a configuration parameter, select the Thing on the left and access the appropriate panel on the right.

### Nodes added, but no items bound

In rare cases a node can be successfully added and configured, but its Z-Wave channels are not bound to any OpenHAB item, so its properties are not accessible from the REST interface or a sitemap. This can also be resolved with HABmin.

Select the Node in the left hand side list, then the Channels section in the right panel. Use the little + button for each channel, and then press Save. The resulting item will have an automatically generated name, type and category, so no further achievements have to be done.

### PST-02A door status not updating

Open the configuration of the device by clicking the pencil icon, then set the Custom Function parameter to 22 to enable sending the door status using the Z-Wave command understood by the binding. Updating configuration values can take a few minutes with the sensor close the controller or can be forced by opening the casing of the device and pressing 3 times the back button 3 times in 1.5 seconds as explained in the sensor documentation.

### Aeotec Multisensor 6 not sending events

The device can take a while to respond, even when in USB power. Try to press the button to trigger a wake, or just wait patiently.

### Aeotec Key Fob Gen5

This device is fairly tricky to get to work in OpenHAB despite being fully supported. Here are the steps I followed:

* Reset the device if the device was previously paired. To do so, hold down the Mode button at the back of the fob for about 20 seconds. The LED should blink red/green, faster and faster until it stays solid green. At this point you can let go of the button.
* In OpenHAB, launch the Z-Wave search, then press button 4 (bottom right) briefly, the LED should go green then blink for a bit. An unknown device should appear in OpenHAB.
* Then, press button 4 for about 3 seconds, the LED should blink green rapidly for several seconds, and the LED on the controller should be blinking as well.
* Refresh the search page, the node should be named appropriately. This means it has been detected properly.
* Add this device
* Next go to Configuration->Things, find the node in the list then click edit.
* Set the mode to Scene mode, Group mode uses the fob as controller rather than input
* Press the tick to save, then hold button 4 for 3 seconds to awake the device and let it receive configuration data. Wait until the LED stops blinking green.
* Switch the device to Use Mode by pressing down both buttons at the back (Mode and Learn) for about 5 seconds (the LED should stay red for about 3 seconds before briefly going green, let go at this moment).
* The key fob should work properly. The scene number should be updated as you press buttons. A table of the scene values is found in the manual of the device.