domogik-plugin-ozwave

Release 0.5.0

May 18, 2017

Contents

1 Plugin documentation **1**
- 1.1 Last changes . . . 1
- 1.2 Purpose . . . 1
- 1.3 Dependencies . . . 2
- 1.4 Zwave Controller and devices Compatibility List . . . 2
- 1.5 Create the udev rule for the Zwave controller . . . 3
- 1.6 Plugin configuration . . . 4
- 1.7 Start the plugin . . . 7
- 1.8 Configure the features widgets on Domoweb . . . 7

2 Openzwave & Manager tools **9**
- 2.1 State and version of libraries . . . 9
- 2.2 Openzwave configuration options . . . 10
- 2.3 Controller and Network tab . . . 10

3 Controller tools **11**
- 3.1 Get Zwave controller informations . . . 11
- 3.2 Drivers and controller commands . . . 12

4 Devices (Nodes) tools **17**
- 4.1 Get Zwave devices (Nodes) informations . . . 17
- 4.2 Manage association devices (groups) . . . 20
- 4.3 Manage devices Commands Class . . . 22
- 4.4 Drivers and controller commands . . . 25

5 Zwave network tools **31**
- 5.1 Graphic neighborhood network . . . 31
- 5.2 Network statistic and tests . . . 33
- 5.3 Node statistic and tests . . . 34
- 5.4 Device test messages . . . 36

6 Support tools **37**
- 6.1 Memory usage . . . 37
- 6.2 Show log file informations . . . 37
- 6.3 List of manufacturers and product compatibilities . . . 38

7 Development informations **39**

7.1 Detailed architecture . . . 39

8 Change log **41**
8.1 0.5.4 : (13-05-2017) . . . 41
8.2 0.5.3 : (27-12-2016) . . . 42
8.3 0.5.2 : (03-08-2016) . . . 42
8.4 0.5.1 : (27-06-2016) . . . 43
8.5 0.5.0 : (11-05-2016) . . . 43
8.6 0.4.0b1 : (12-10-2015) Compatibily library : OpenZwave >= 1.3.401, 0.3.0 (b6) <= python_openzwave >= 0.3.0 (b4) . . . 44
8.7 0.4.0a1 : (30-03-2014) Compatibily library : OpenZwave >= 1.0.711 (/branches/2013-11-13_release_testing), python_openzwave >= 0.2.5 (rev >= 3bef0f1cb27f) . . . 44
8.8 Do an insert data . . . 47

9 Advanced - Dependencies installation **49**
9.1 Install the tailer library for Python . . . 49
9.2 Install python-openzwave . . . 49

10 Advanced - Usage of udev rules **55**
10.1 Create the udev rule for controller . . . 55

CHAPTER 1

Plugin documentation

Last changes

Before installation a new release of this plugin, please check the *changelog*.

Purpose

Note: Please notice that this plugin is **still in development**!

If you find any issue, please create a ticket on the Gihub repository : https://github.com/Nico0084/domogik-plugin-ozwave/issues

In the same way, if something is not clear or wrong in this documentation, feel free to open a ticket!

Z-Wave is a wireless ecosystem that lets all your home electronics talk to each other, and to you, via remote control. This plugin allows to control zwave devices.

It uses the open source library openZwave c++ project and python-openzwave as interfacing cython,

The Zwave network manager is directly integrated into the plugin.

Simple action/sensor of devices have access via domogik devices (widgets). Viewing and setting Zwave devices is accessed via a special plugin page from the admin panel.

Development is in progress, features will get gradually added.

Steps to set up your first Zwave device

To set up your first Zwave device, you will have to :

- install this plugin on domogik
- install this plugin dependencies
- create an udev rule for your Zwave controller (the usb device you plug on the computer)
- configure this plugin
- create a Domogik device for your Zwave controller
- start the plugin
- in the *Advanced* pages, look for your Zwave device and get informations about it
- create the Domogik device for your Zwave device

Dependencies

- Python-openzwave (>=0.3.1)
- tailer 0.3 library for Python (>=0.2.1)

If ou are using a **Debian based** Linux release, you can install the dependencies with an installation script (in the root of the ozwave package folder) :

```
$ sudo ./install_dependencies.sh
```

Options :

- -LAST : get the last python-openzwave archive from Git repository.
- -v x.x.x : get a specific python-openzwave archive from Git repository.
- nothing : get default python-openzwave archive from Git repository.

Else, you can *follow the detailed instructions*.

Zwave Controller and devices Compatibility List

The following controller interfaces are supported and verified with Domogik:

- Aeon Labs Z-Stick Series 2
- Aeon Labs Z-Stick Gen5
- RaZberry * using rule with /dev/ttyAMA0 instead of /dev/ttyUSBx) * user domogik must have write permission, you can add it in dialout group.

  ```
  $ sudo usermod -a -G dialout domogik
  ```

Other controllers are also supported by openzwave, you can check here

The following devices are supported :

- Everspring
  - ST814 - Temperature, Humidity Sensor

  - AN158 - Switch Meter Plugin
  - SE812 - Siren
- Everspring (C.T.)
  - HSM02 - Door windows sensor
- Fibaro
  - FGS211 - Relay Switch 3KW
  - FGS221 - Double Relay Switch 2x1,5kW
  - FGD211 - Universal Dimmer 500W
  - WallPlug - Meter Switch with leds
- Aeon Labs
  - HEM - Home Energy Metter
  - DSB05 - Motion Multi Sensor
  - ZW089 - Recessed Door Sensor Gen5
- Express Controls
  - HSM100 - EZMotion luminosity and temperature sensor
- Danfoss
  - Living connect (thermost heating)
- Qubino
  - ZMNHCDx - Flush Shutter

Create the udev rule for the Zwave controller

You may create a udev rule for this device. The rule and filename are given in the **Requirements** tab of the plugin configuration page.

Currently, your PC controller is known as **/dev/ttyUSBx** (by default). This is not very convenient nor meaningful.

We will so create a new udev rule that will create a link called **/dev/zwave** that will point to **/dev/ttyUSBx**.

Zwave controller model	Udev rule file
Aeon Labs Gen4 Zwave	98-usbcp210x.rules
Aeon Labs Gen5 Zwave+	99-usbzstickgen5.rules

To install a udev rule, copy the appropriate file in the udev rules folder on your system. Example

```
$ sudo cp udev/98-usbcp210x.rules /etc/udev/rules.d/
```

Then, you can use the following command to apply the udev rule, or unplug/plug the Zwave controller.

```
$ sudo udevadm test $(udevadm info --query path --name ttyUSB0)
```

If your controller is not handled by any of the proposed udev rules, please check the *tutorial on how to create your own udev rule*.

Plugin configuration

Configuration

In Domogik administration section, go to the plugin ozwave configuration page.

Key	Type	Description
autoconfpath	boolean	If checked, the plugin will try to automatically find the Openzwave library path. If not checked, you will have to manually set the Openzwave library path with the parameter **configpath**.
configpath	string	If **autoconfpath** is not checked, set up the Openzwave library path. Example : `/usr/local/lib/python2.7/dist-packages/libopenzwave-0.3.0b7-py2.7-linux-x86_64.egg` In this folder you will find all the Openzwave xml files.
cpltmsg	boolean	Notifications when transaction complete is reported. **Note:** **In any case the value of openzwave option.xml** If **NotifyTransactions** is enable in your option.xml comment it to get control by plugin.
ozwlog	boolean	If checked, activate the Openzwave logger. Please do this only for debug purpose as the logs are very verbose! **The file is located in the data folder of the plugin and** **Note:** **In any case, if exist, the value of option.xml file** If **Logging** is enable in your option.xml comment it to get control by plugin.

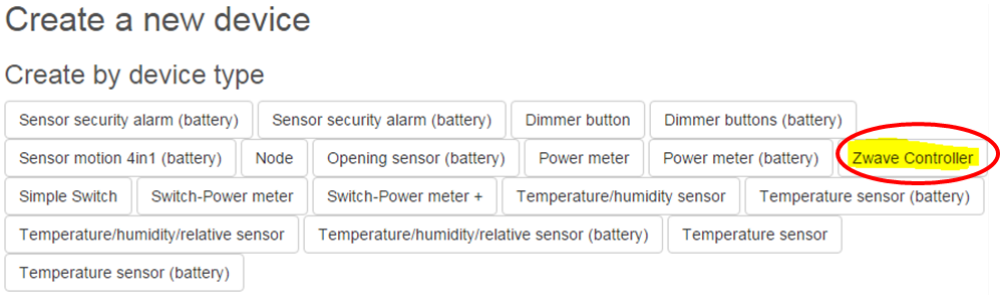
Now, you will need to create a device for your Zwave controller...

Create a device for the primary Zwave controller

In clients page of admin UI, go to the **Domogik devices** page.

To create a device, click on **Create a new Domogik device**

Then, select the device type : **Zwave Controller**.



This device will create a link between the controller device and Openzwave. It will also create a link between the **Zwave homeId** and your **networkId** used by Domogik for the devices adresses.

You can create the controller device after starting the plugin, so it is possible to find its address in the devices table of the **controller and devices**.

You can create as many **Zwave Controler** you have primary controllers.

Note: Creating this device is necessary to handle your zwave network.

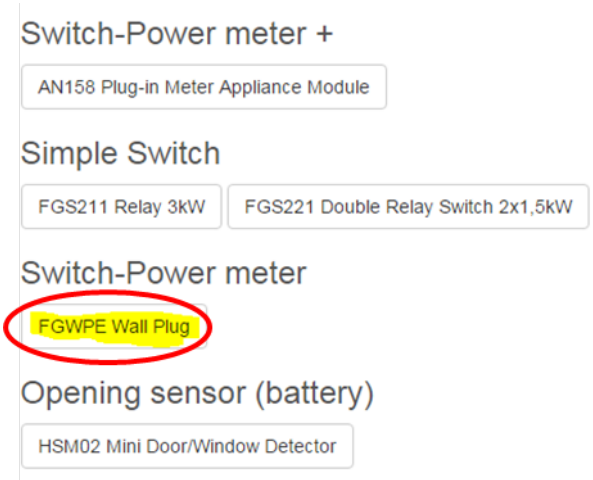
Main parameters :

Key	Example	Description
Device Name	MyController	The display name for this device.
Description	What you want	A short descriptionn for this device.
Reference	Z-Stick 2	A reference for this device, eg.: manufacturer reference.

Global parameters :

Key	Example	Description
driver	/dev/zwave	Z-wave device that you have define in UDEV rule
networkid	MyHomeNetwork	Home ID name for association with domogik devices

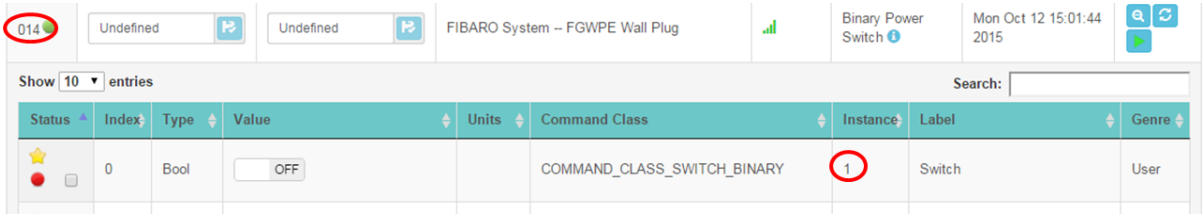
Create a device for a Zwave device



Main parameters :

Key	Example	Description
Device Name	MyController	The display name for this device.
Description	What you want	A short descriptionn for this device.
Reference	FGWPE Wall Plug	A reference for this device, eg.: manufacturer reference.

Global parameters :



Key	Example	Description
networkid	MyHomeNetwork	Home ID name for association with domogik devices
node	14	Z-wave node id that you can find in the nodes table
instance	1	Zwave node instance id that you can find in the Commands Class table

Extra parameters for some particular zwave device :

Key	Example	Description
batterycheck	True	HCheck battery level at zwave device wakeup.

Start the plugin

You can now start the plugin (start button) and use the created devices.

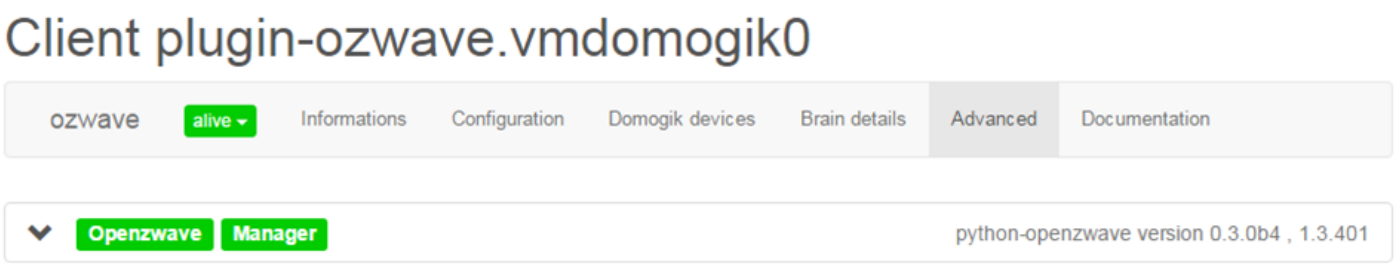
Configure the features widgets on Domoweb

You can now place the widgets of your devices features on Domoweb.

CHAPTER 2

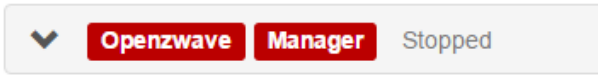
Openzwave & Manager tools

State and version of libraries



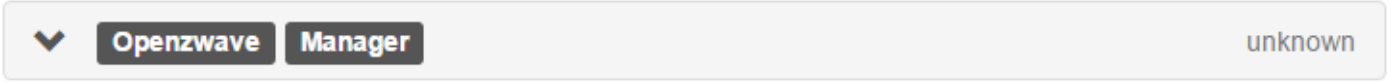
The two flag **Openzwave** and **Manager** give the state of respective library :

- Red : Stopped



- Orange : Starting
- Green : Alive

- Grey : Unknown
- Black : Failed

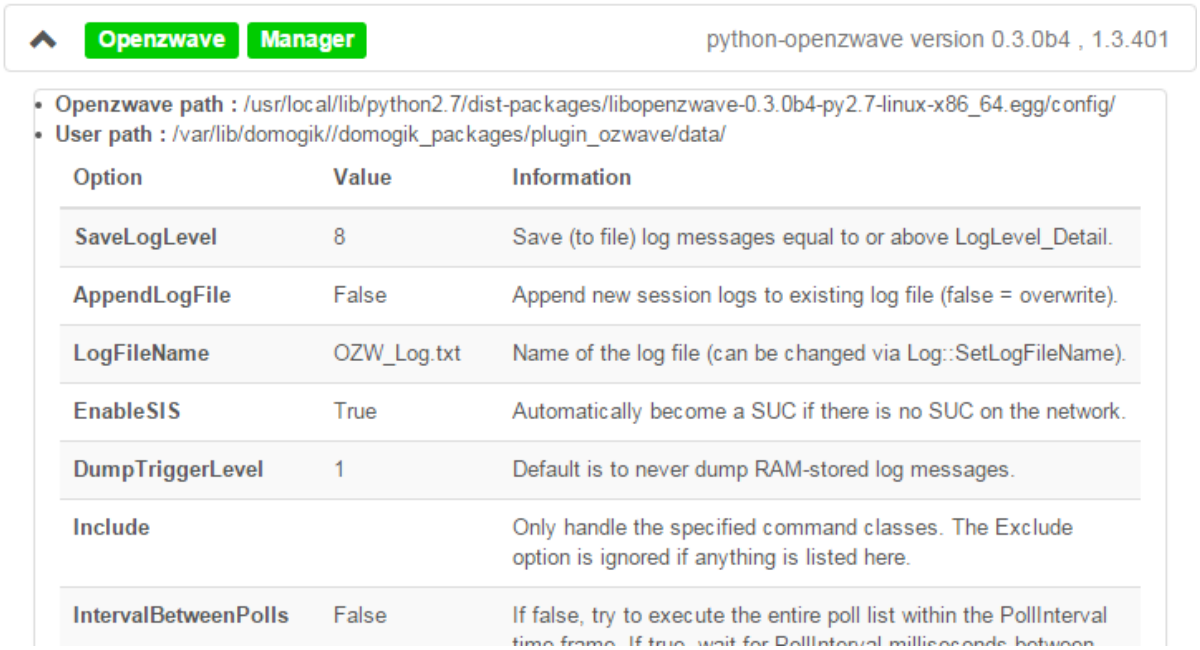


Openzwave: is the openzwave librarie installed. Manager: is the ozwave plugin manager connected to openzwave.

On right python-openzwave and openzwave libraries version are display, only if the plugin is started.

This row can be collaspe to display openzwave configuration options.

Openzwave configuration options

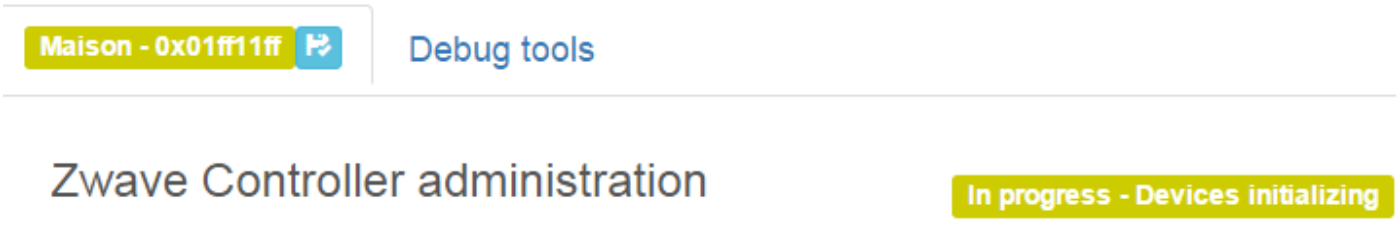


Here you can check all options. All details are commented in table.

Soon in a future release, it may be possible to edit them..

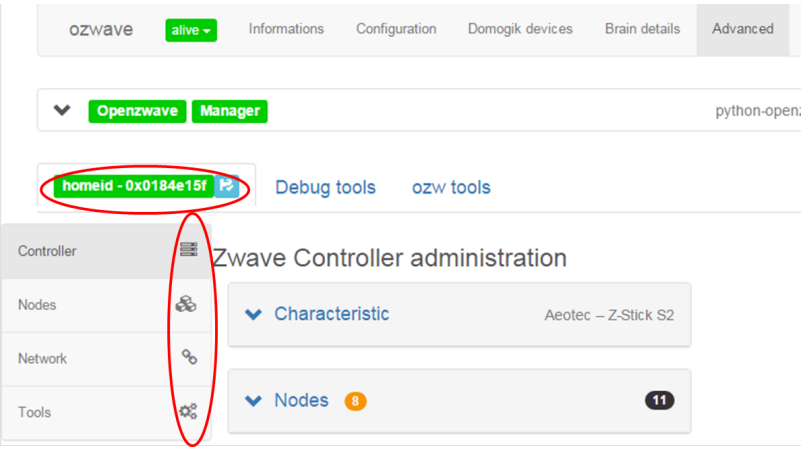
Controller and Network tab

Under state libraries row, a tab list and display all controller knowns.



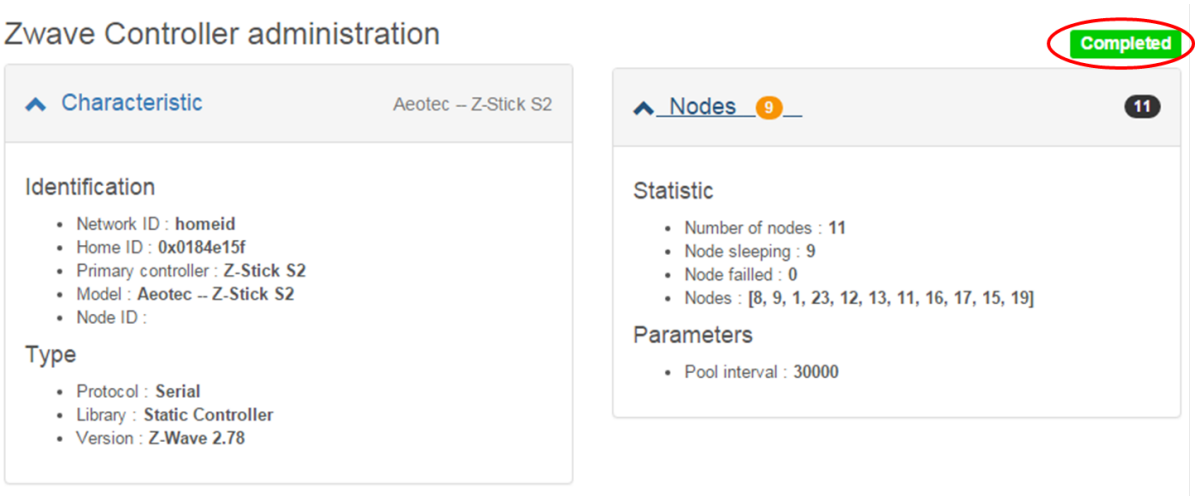
CHAPTER 3

Controller tools



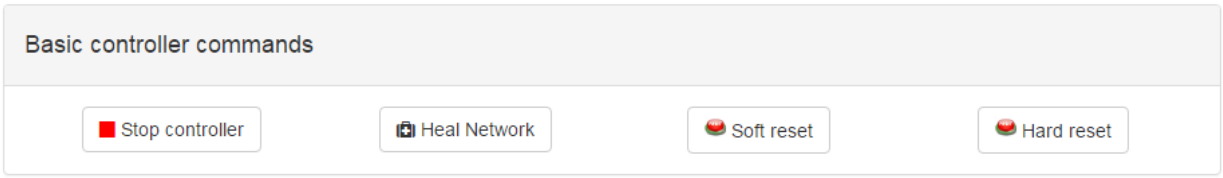
Get Zwave controller informations

- you can check to verify the information on zwave network state and controller in the first part of the page.



Drivers and controller commands

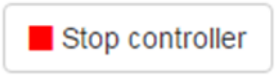
In this section, orders can be sent directly to the driver



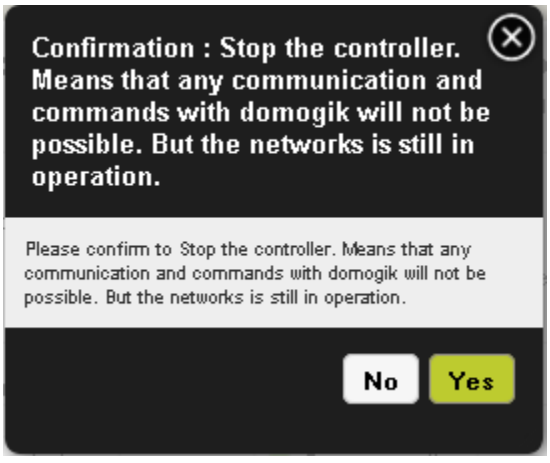
Stop and Start driver (controller)

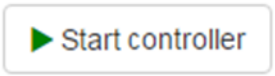
TODO :

Driver automatically starts and start/stop button automatically changes depending on the state of the driver.

- You can stop it when it's possible : 

confimation dialog aks you to continue.

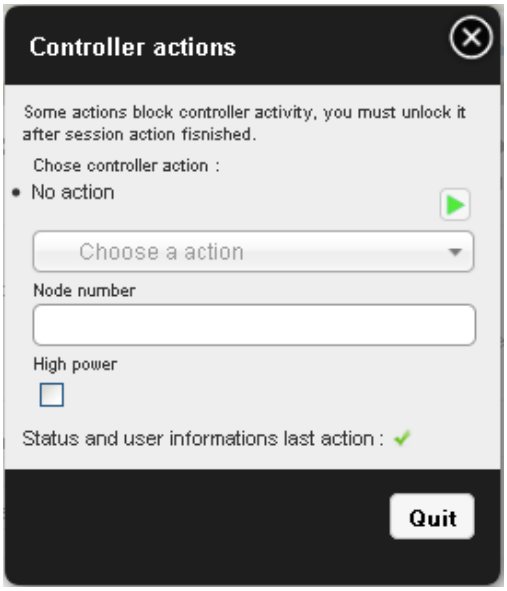


- You can Start it when it's possible : 

Initialization process running, you must be patient ....

Controller commands and actions

- You access to dialog box by button



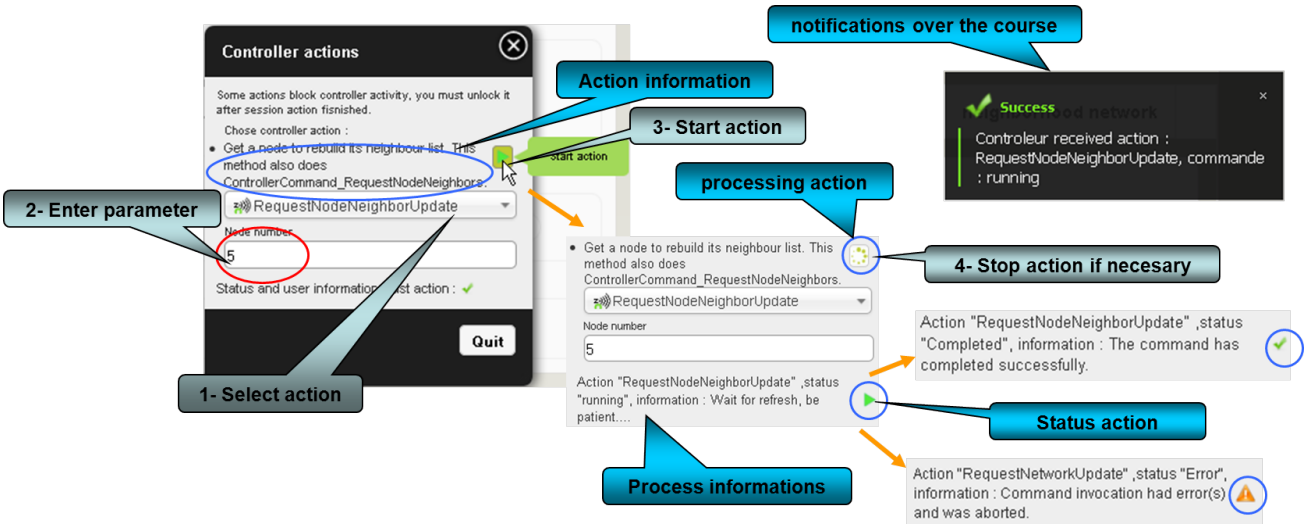
Actions process

- This modal dialog contains informations to guide the user.
- Just actions "CreateButton" and "DeleteButton" are not implemented yet.

Action / command	Description
AddDevice	Add a new device (but not a controller) to the Z-Wave network.
CreateNewPrimary	Add a new controller to the Z-Wave network. The new controller will be the primary, and the current primary will become a secondary controller.
ReceiveConfiguration	Receive Z-Wave network configuration information from another controller.
RemoveDevice	Remove a new device (but not a controller) from the Z-Wave network.
RemoveFailedNode	Move a node to the controller's failed nodes list. This command will only work if the node cannot respond.
HasNodeFailed	Check whether a node is in the controller's failed nodes list.
ReplaceFailedNode	Replace a non-responding node with another. The node must be in the controller's list of failed nodes for this command to succeed.
TransferPrimaryRole	Make a different controller the primary.
RequestNetworkUpdate	Request network information from the SUC/SIS.
RequestNodeNeighborUpdate	Get a node to rebuild its neighbour list. This method also does ControllerCommand_RequestNodeNeighbors.
AssignReturnRoute	Assign a network return routes to a device.
DeleteAllReturnRoutes	Delete all return routes from a device.
SendNodeInformation	Send a node information frame.
ReplicationSend	Send information from primary to secondary.
CreateButton	Create an id that tracks handheld button presses.
DeleteButton	Delete id that tracks handheld button presses.

Schematic processing

- Each action have differents step, some will attempt user action on device, some will take a long time, some must be stop manually.
- You can stop action, but stop result have some time a fail report.

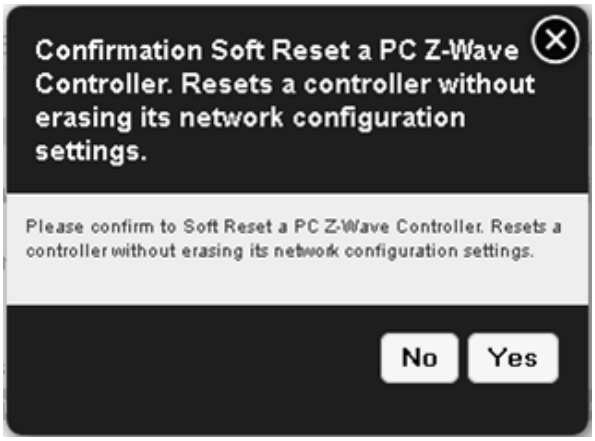


Reset controller

- Two mode possible with acces by buttons :

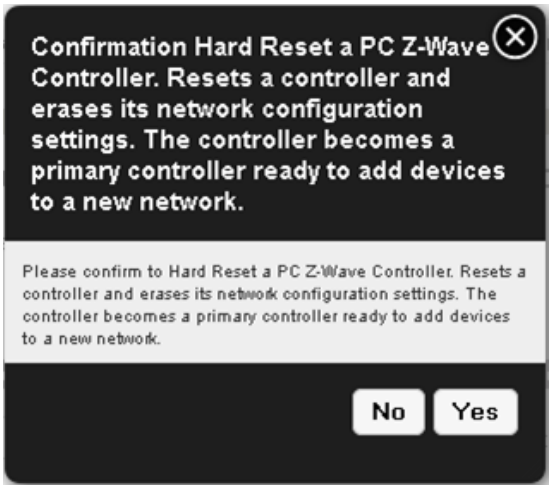
Soft reset

- Soft Reset a PC Z-Wave Controller who resets a controller without erasing its network configuration settings



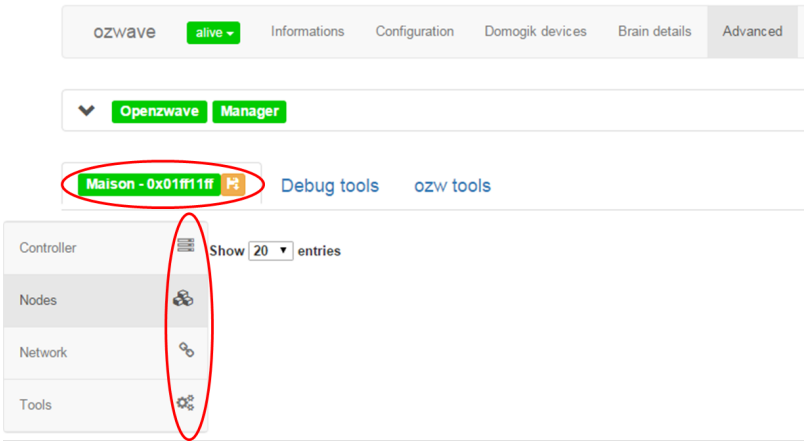
Hard reset

Warning: Be careful this action is irreverssible



CHAPTER 4

Devices (Nodes) tools

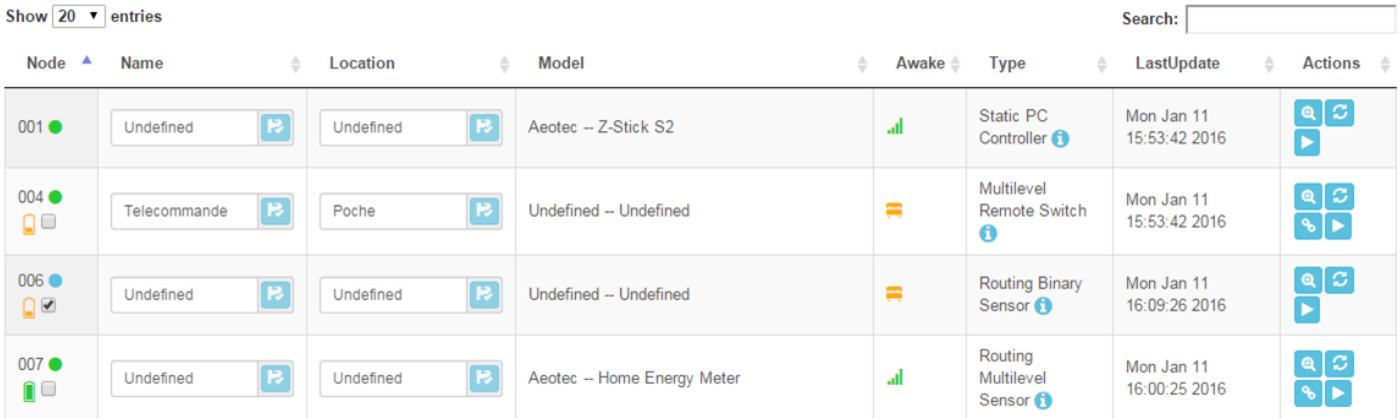


Get Zwave devices (Nodes) informations

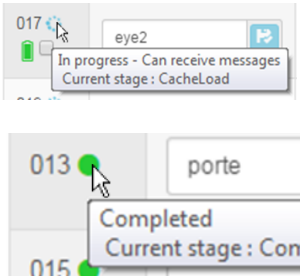
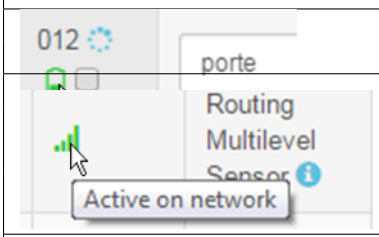
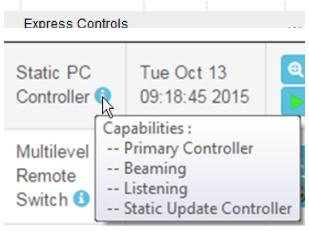
Note: On a zwave system, devices are called **Nodes**.

- **Devices initialisation** field give state of global initialize process.

The process can take a long time (up to 5 min). Be patient..... All nodes discovered and / or stored are scanned one by one to be initialized.

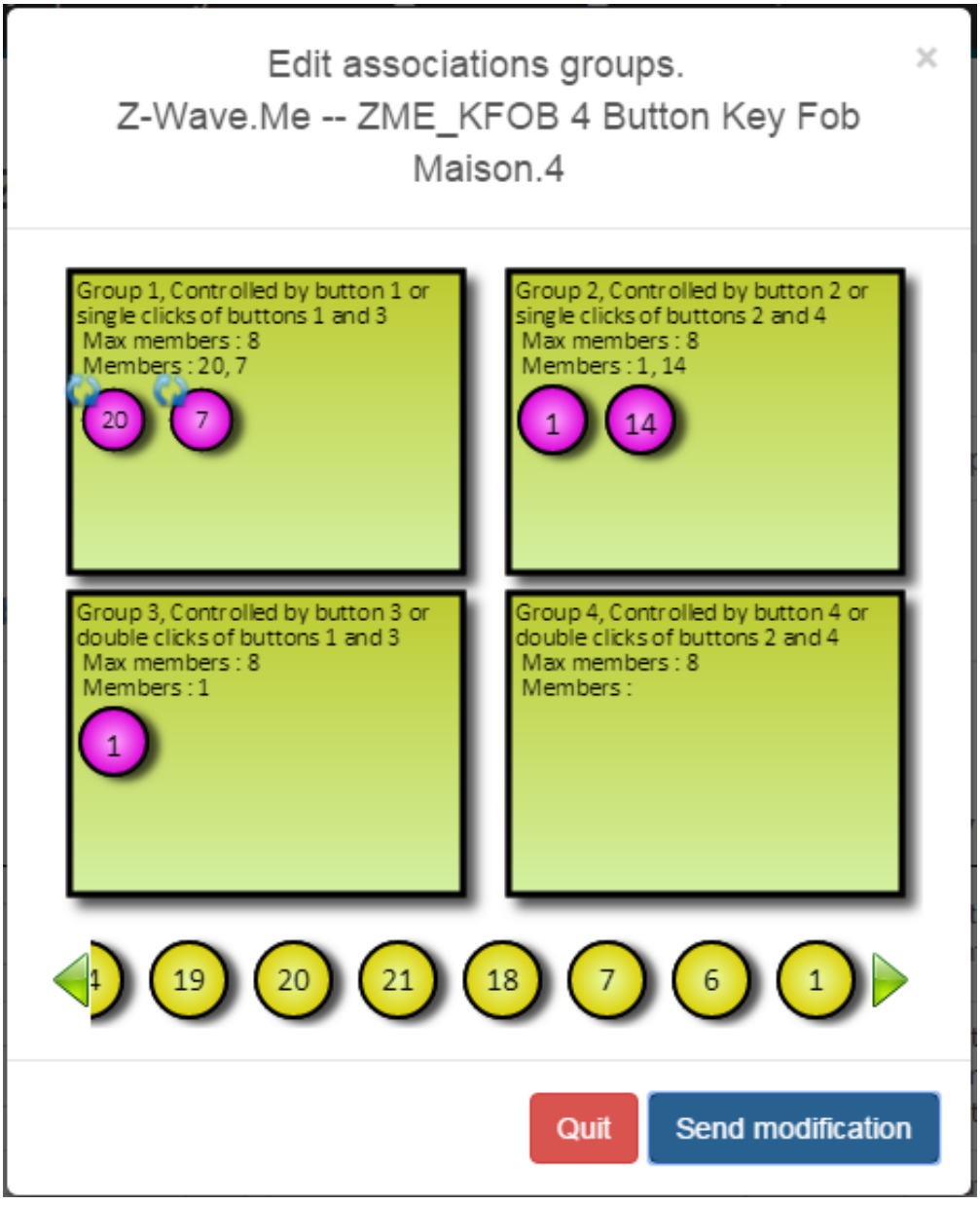


- **Detail status information**.

Representation	Message	Description
	**Pointing the mouse over icon in column "NodeId" give status of initializing.**	
	Uninitialized	No starting init operation.
	Initialized - not known	Node has completed init, but open-zwave don't know is model (shows xml openzwave files).
	Completed	Node has full completed init.
	In progress - Can receive messages (Not linked)	Node can now receive messages, but his initialization not finish. Probably node is sleeping Controller have finish init, when node awake it state should be Completed.
	In progress - Linked to controller	Node is recognize by controller, initialization is in progress.
	In progress - Can receive messages	Node can now receive messages, but initialization not finish.
	Out of operation	Node is mark as failed, controller don't find him.
	**Pointing the mouse over battery icon in column "NodeId" give status of battery level.**	
	Value in %	Fill icon picture level of battery.
	Pointing the mouse over icon in column "Awake" give status of listening mode.	
	Awake	Node is awake, message can send immediately.
	Sleeping	Node probably sleeping, message are put on send queue.
	**Pointing the mouse over icon in column "Type" give capacities of node.**	
	Primary Controller	Node is primary controller, is the main device used to configure and control a Z-Wave network. The only difference between a primary and secondary controller is that the primary is the only one that can be used to add or remove other devices. For this reason, it is usually better for the primary controller to be portable, since most devices must be added when installed in their final location.
	Secondary Controller	There can only be one primary controller - all other controllers are secondary controllers.
	Static Update Controller	A Static Update Controller (SUC) is a controller that must never be moved in normal operation and which can be used by other nodes to receive information about network changes.
	Bridge Controller	A bridge controller is able to create

Manage association devices (groups)

- Some devices have the ability to be combined with other devices and can send their information.
- These associations are using groups, access to the dialog management groups is done with the button



how to use management associations

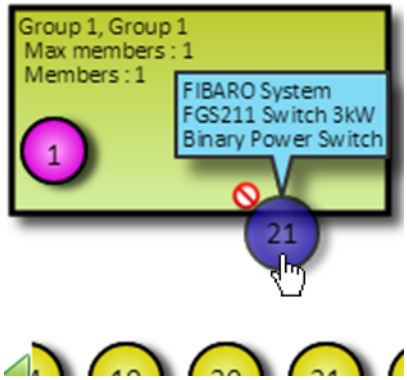
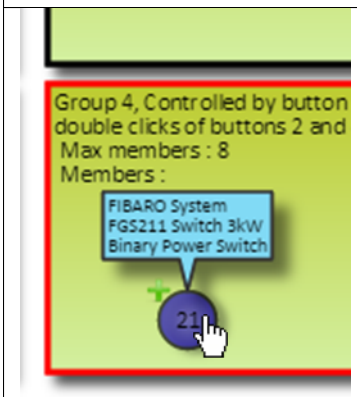
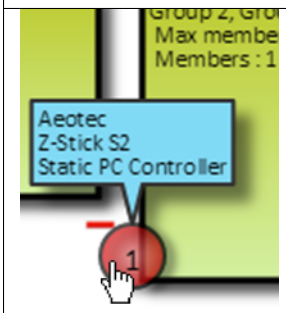
Note:

- Due to the possibility of sleeping devices, the system considers the command to have been transmitted.
- In the case of a device sleeping command will be transmitted when the device wakes up.

- So the association will be effective as this momment there.

Warning: Careful if the plugin is stopped between time command is lost.

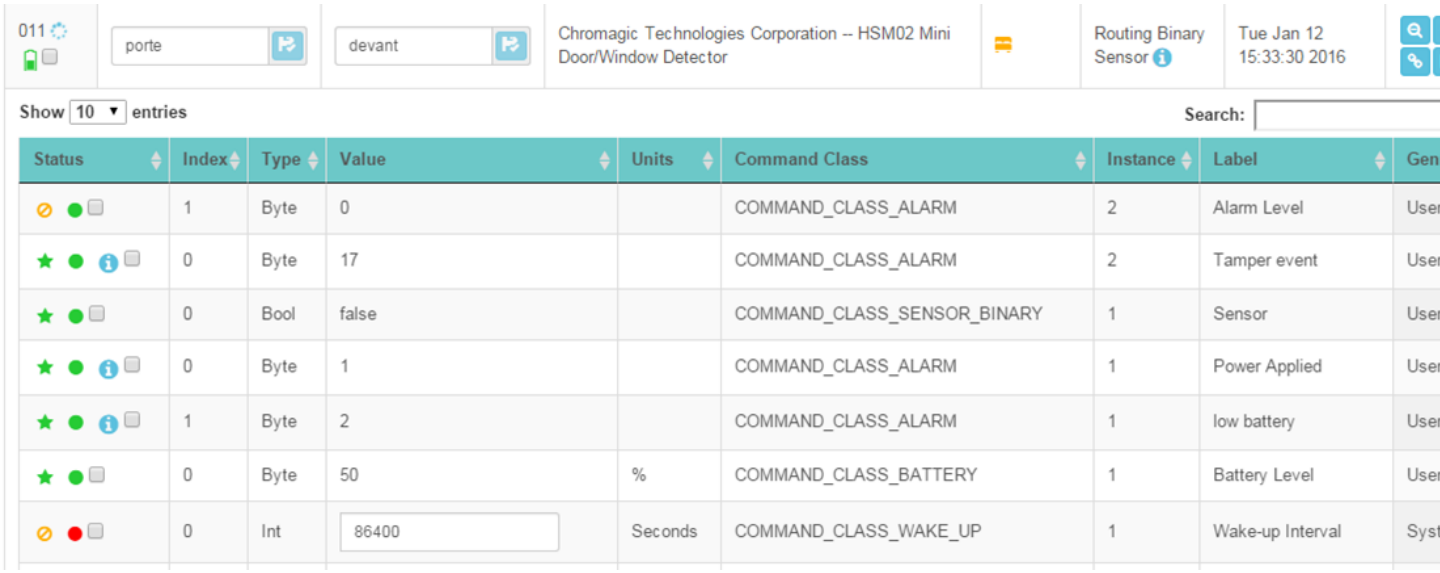
- An icon indicates the status of the device in the group, it may be :
  - unknown
  - confirmed
  - to confirm
  - to update
- The operations are performed by simply drag and drop.
- After making the changes click on **OK** to send at the device.
- To quit dialog box click **Cancel**.
- Actually button **Reset** is not handled.

Example	Description
	If you drop at bad placement a icon forbidden is show.
	When you drop in a group who device ins'it to add it, a icon plus is show.
	After adding device in group, a icon to update is show.
	If you drop at bad placement a icon forbidden is show.

Manage devices Commands Class

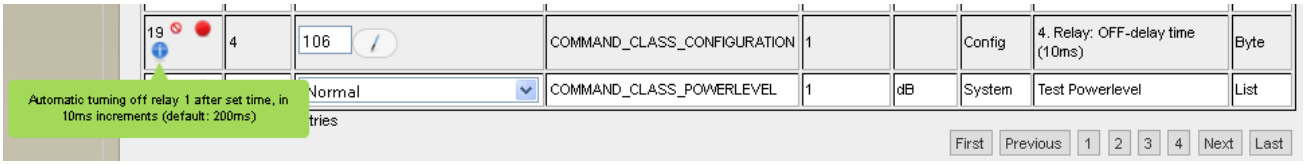
Display commands class list

- All command class are listed in a table.

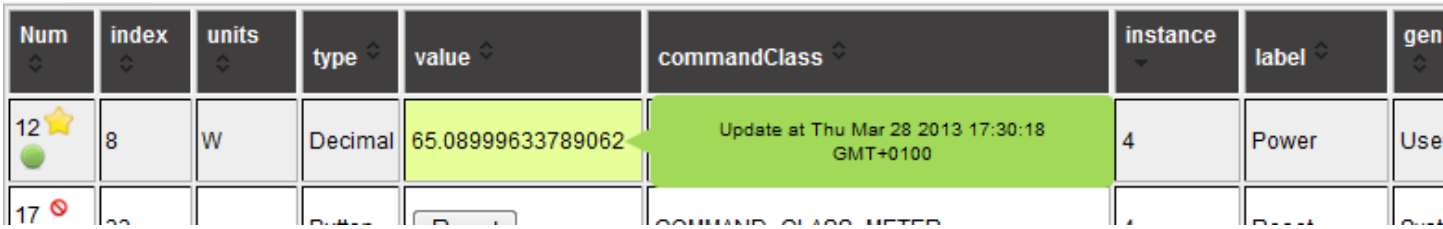


Change values of command class

- You can edit the values that are in writing, if the value is changed one “Edit” button appears, you must click it to send change at device
- If exist, pointing the mouse over icon “i” give information about command-class.



- When a value of command class is updated, she his tagged during 5 secondes. Pointing the mouse over value give date of update.



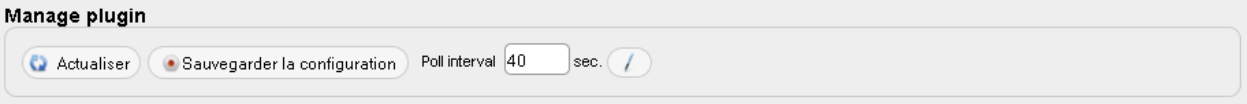
Poll service

- Modern devices will not require polling. Some old devices need to be polled as the only way to detect status changes.
- Due to patent concerns, some devices do not report state changes automatically to the controller.

These devices need to have their state polled at regular intervals. The length of the interval is the same for all devices. To even out the Z-Wave network traffic generated by polling, OpenZWave divides the polling interval by the number of devices that have polling enabled, and polls each in turn. It is recommended that if possible, the interval should not be set shorter than the number of polled devices in seconds (so that the network does not have to cope with more than one poll per second).

Set time interval

- Set in seconds global interval, click send button to confirm.

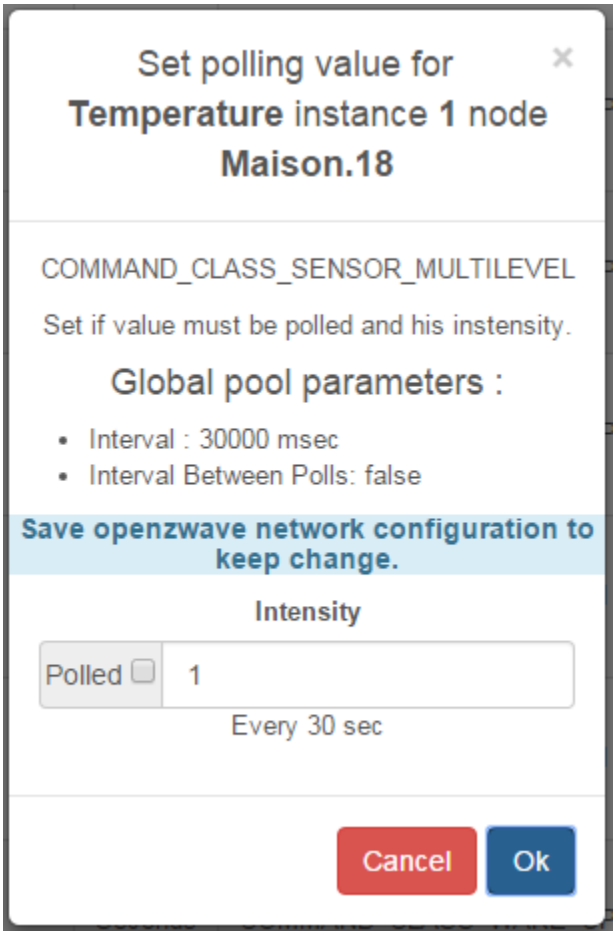


Polling a command class

- In first column a check box give access to poll value.

Num	index	units	type	value	commandClass	instance	label	gen
11	3	%	Decimal	89	COMMAND_CLASS_SENSOR_MULTILEVEL	2	Luminance	User
12	1		Byte	200	COMMAND_CLASS_CONFIGURATION	1	Sensitivity	Confi
13	1	°C	Decimal	21.83333502875434	COMMAND_CLASS_SENSOR_MULTILEVEL	3	Temperature	User
14	1	Seconds	Int	360	COMMAND_CLASS_WAKE_UP	1	Minimum Wake-up Interval	Syste

- A dialog box appeared to confirme action. Here check if polled and set intensity, the number (frequency) of poll during global interval.

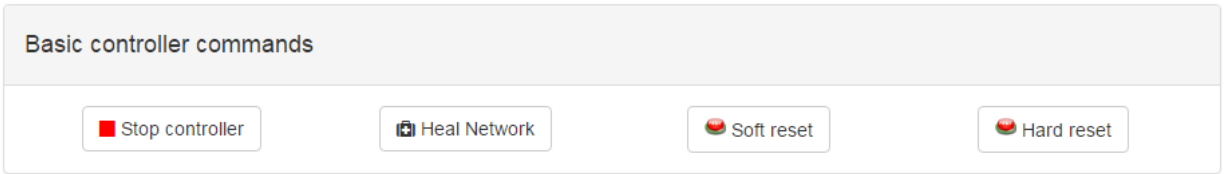


Warning:

- Polling sleeping devices put openzwave library in waiting confirmation. This could raise a network error.
- Using an intensity value more 2 could raise a network error.

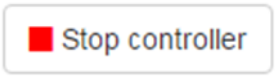
Drivers and controller commands

In this section, orders can be sent directly to the driver

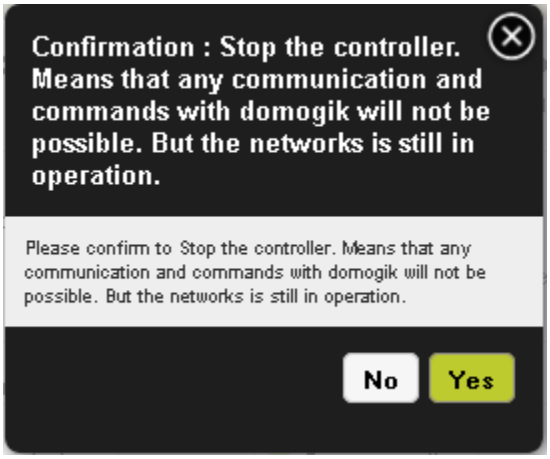


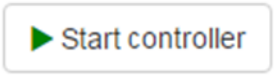
Stop and Start driver (controller)

Driver automatically starts and start/stop button automatically changes depending on the state of the driver.

- You can stop it when it's possible : 

confimation dialog aks you to continue.

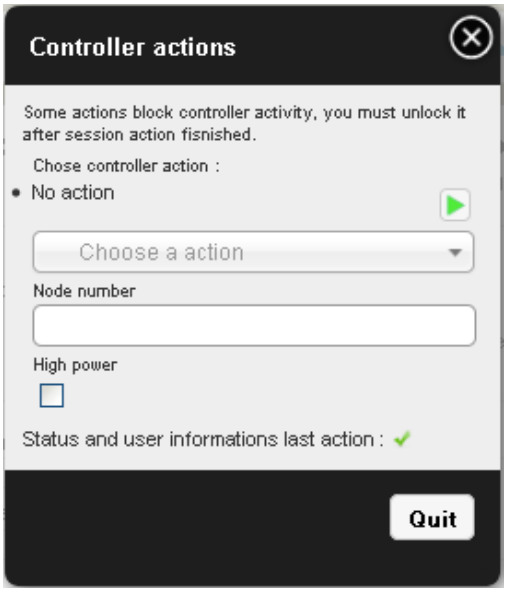


- You can Start it when it's possible : 

Initialization process running, you must be patient ....

Controller commands and actions

- You access to dialog box by button



Actions process

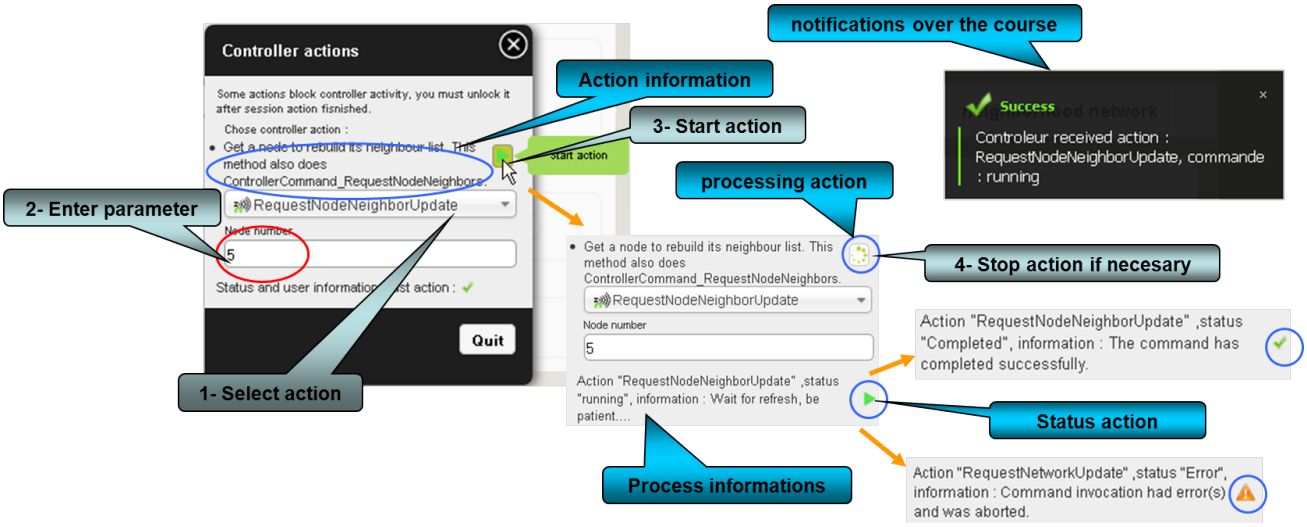
- This modal dialog contains informations to guide the user.

- Just actions “CreateButton” and “DeleteButton” are not implemented yet.

Action / command	Description
AddDevice	Add a new device (but not a controller) to the Z-Wave network.
CreateNewPrimary	Add a new controller to the Z-Wave network. The new controller will be the primary, and the current primary will become a secondary controller.
ReceiveConfiguration	Receive Z-Wave network configuration information from another controller.
RemoveDevice	Remove a new device (but not a controller) from the Z-Wave network.
RemoveFailedNode	Move a node to the controller’s failed nodes list. This command will only work if the node cannot respond.
HasNodeFailed	Check whether a node is in the controller’s failed nodes list.
ReplaceFailedNode	Replace a non-responding node with another. The node must be in the controller’s list of failed nodes for this command to succeed.
TransferPrimaryRole	Make a different controller the primary.
RequestNetworkUpdate	Request network information from the SUC/SIS.
RequestNodeNeighborUpdate	Get a node to rebuild its neighbour list. This method also does ControllerCommand_RequestNodeNeighbors.
AssignReturnRoute	Assign a network return routes to a device.
DeleteAllReturnRoutes	Delete all return routes from a device.
SendNodeInformation	Send a node information frame.
ReplicationSend	Send information from primary to secondary.
CreateButton	Create an id that tracks handheld button presses.
DeleteButton	Delete id that tracks handheld button presses.

Schematic processing

- Each action have differents step, some will attempt user action on device, some will take a long time, some must be stop manually.
- You can stop action, but stop result have some time a fail report.

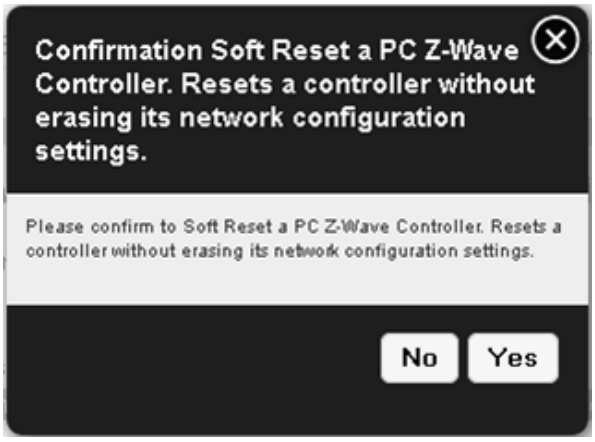


Reset controller

- Two mode possible with acces by buttons :

Soft reset

- Soft Reset a PC Z-Wave Controller who resets a controller without erasing its network configuration settings



Hard reset

Warning: Be careful this action is irreverssible

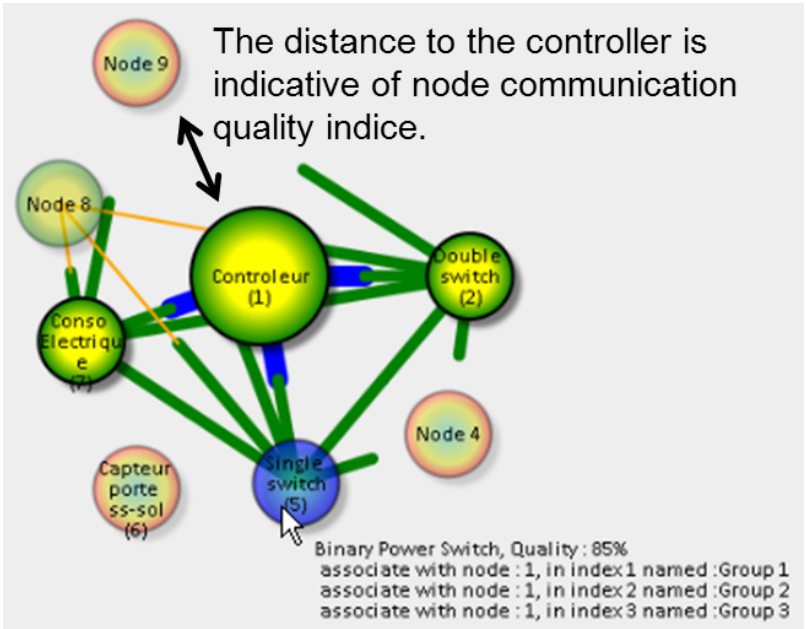
Confirmation Hard Reset a PC Z-Wave Controller. Resets a controller and erases its network configuration settings. The controller becomes a primary controller ready to add devices to a new network.
Please confirm to Hard Reset a PC Z-Wave Controller. Resets a controller and erases its network configuration settings. The controller becomes a primary controller ready to add devices to a new network.
No
Yes

CHAPTER 5

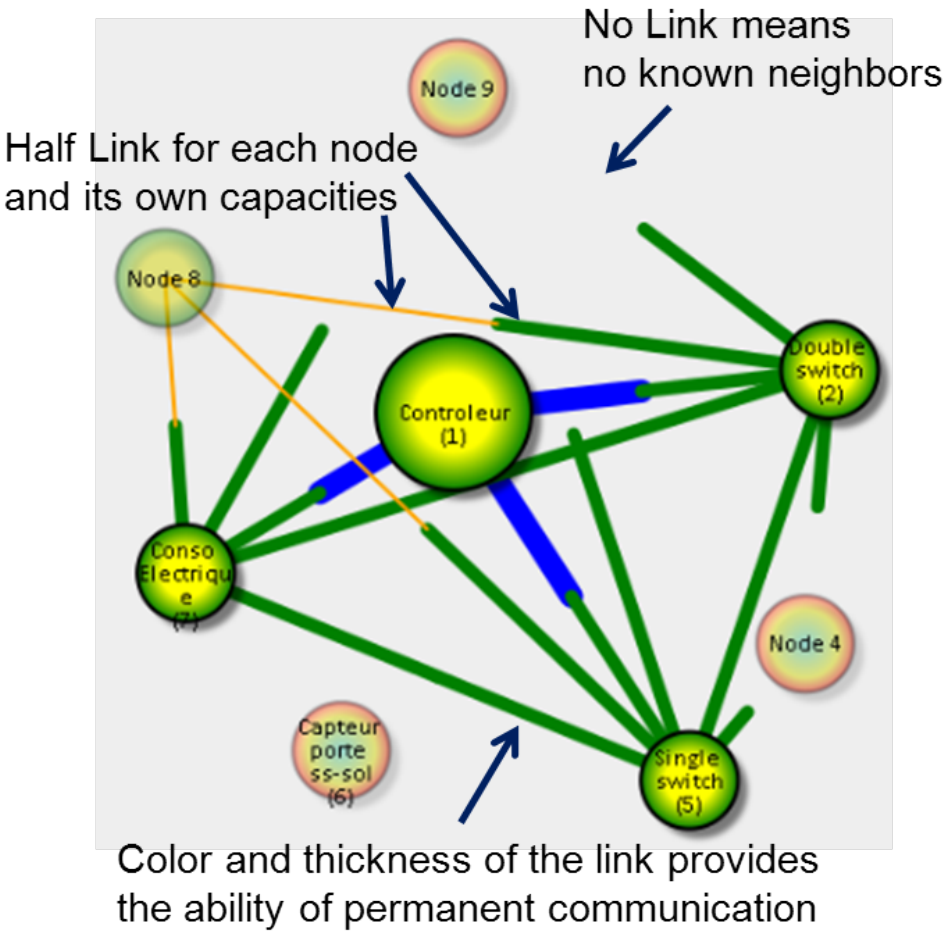
Zwave network tools

Graphic neighborhood network

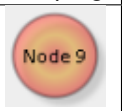
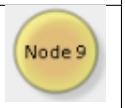
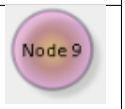
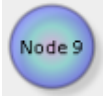
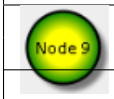
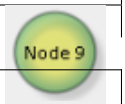
- You can refresh graphic by button



- You can move nodes and arrange it for better links view, but not function to save position for the moment.


No Link means
no known neighbors
Half Link for each node
and its own capacities
Node 9
Node 8
Controleur
(1)
Double
switch
(2)
Conso
Electriqu
e
(7)
Node 4
Capteur
porte
ss-sol
(6)
Single
switch
(5)
Color and thickness of the link provides
the ability of permanent communication

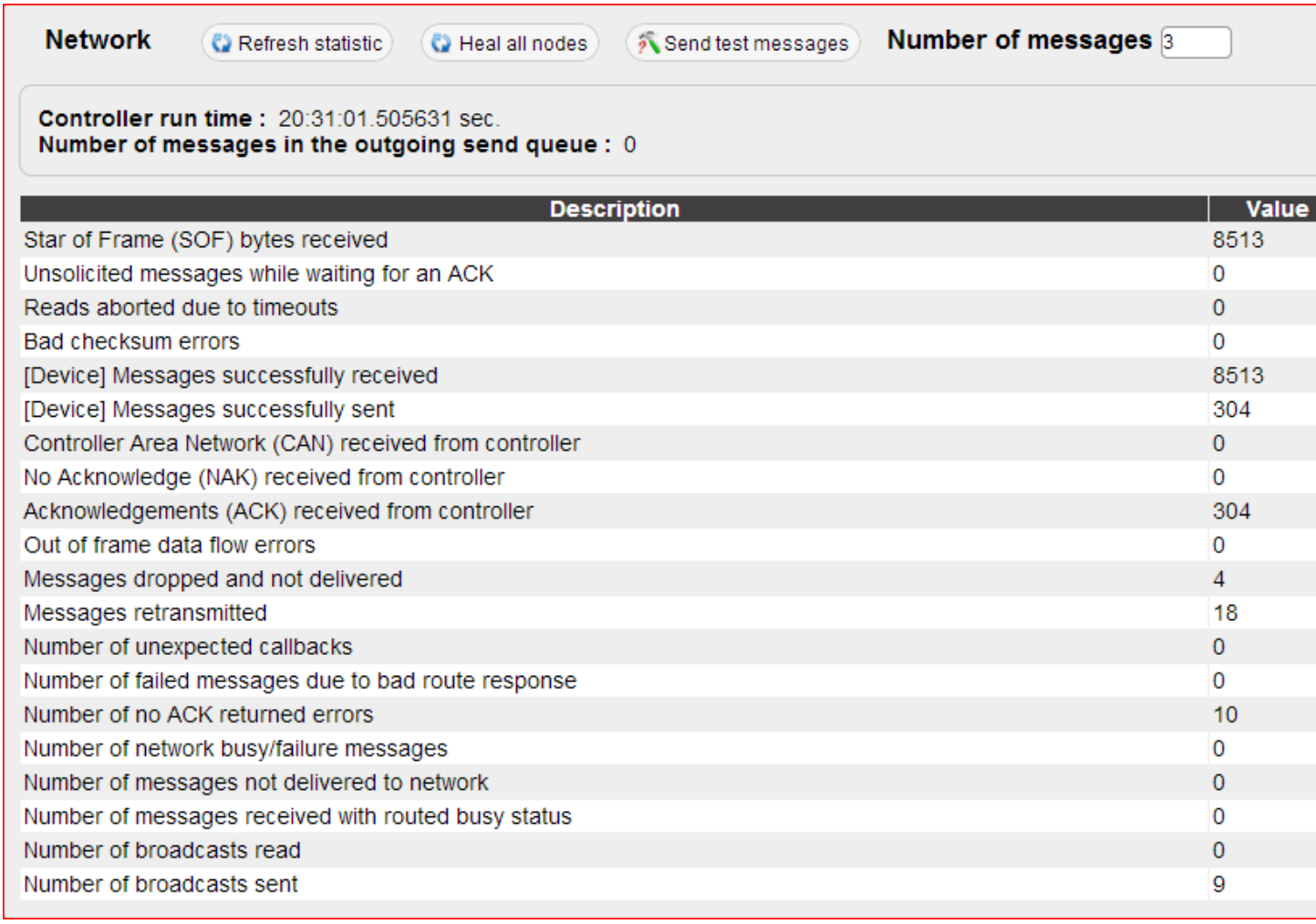
Representation of the node according to the initialisation state.

Node awake	Init state	node sleeping
	Uninitialized	
	Initialized - not known	
	In progress - Devices initializing	
	In progress - Linked to controller	
	In progress - Can receive messages	
	In progress - Can receive messages (Not linked) Means that node probably sleeping during init process, at his wake-up initialization will be completed.	
	Completed	
	Out of operation (failed)	

Network statistic and tests

Tree actions possible :

- Get information about network statistic.
- Start a healing process with reroute all node (node byn node).
- Send test message for all node awake (not sleeping node) - Enter number of send test message per node.



Description	Value
Star of Frame (SOF) bytes received	8513
Unsolicited messages while waiting for an ACK	0
Reads aborted due to timeouts	0
Bad checksum errors	0
[Device] Messages successfully received	8513
[Device] Messages successfully sent	304
Controller Area Network (CAN) received from controller	0
No Acknowledge (NAK) received from controller	0
Acknowledgements (ACK) received from controller	304
Out of frame data flow errors	0
Messages dropped and not delivered	4
Messages retransmitted	18
Number of unexpected callbacks	0
Number of failed messages due to bad route response	0
Number of no ACK returned errors	10
Number of network busy/failure messages	0
Number of messages not delivered to network	0
Number of messages received with routed busy status	0
Number of broadcasts read	0
Number of broadcasts sent	9

Node statistic and tests

two actions possible :

- Get information about a node statistic - Enter node number and click button refresh.
- Send test message for all node awake (not sleeping node) - Enter number of send test message per node.

Note: To heal a specific node use button on columm “Action” of nodes table .

Node statistic : 5 Refresh statistic Send test messages Number of messages 3

Node 5 success test 1/3 in 666 ms.
Node 5 success test 2/3 in 909 ms.
Node 5 success last test 3 in 894 ms, all tests in 2475 ms.

Description	Valeur
Number of messages sent from this node.	32
Number of sent messages failed	0
Number of message retries	0
Number of messages received from this node.	27
Number of duplicated messages received.	0
Number of messages received unsolicited.	0
Last message sent time.	2013-03-27 21:28:48:489
Last message received time	2013-03-27 21:28:48:515
Average Request Round Trip Time (ms).	17
Last message response RTT	26
Average Reponse round trip time.	25
Node quality measure	0
Last received raw data message	00 04 00 02 05 70 06 2b 01 01 aa 7f 00 00 01 80 ad fb 00 00 00 00 d0 7b c8 4e 66 7f 00 00 d0 7b c8 4e 66 7f 00 00 d0 7b c8 4e 66 7f 00 00 d0 7b c8 4e 66 7f 00 32 d4 7b c8 4e 66 7f 00 00 d7 7b c8 4e 66 7f 00 00 d0 7b c8 4e 66 7f 00 00 d7 7b c8 4e 66 7f 00 00 00 00 00 00 00 00 00 00 18 00 00 00 30 00 00 00 10 7d c8 4e 66 7f 00 00 50 7c c8 4e 66 7f 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 38 4a b7 02 00 00 00 00 10 7a c8 4e 66 7f 00 00 20 7a c8 4e 66 7f 00 00 31 85 34 52 66 7f 00 00 00 00 00 00 00 00 00 00 38 4a b7 02 00 00 00 00 50 7a c8 4e 66 7f 00 00 40 7a c8 4e 66 7f 00 00 60 7a c8 4e 66 7f 00 00 d2 83 34 52 66 7f 00 00 00 00 00 00 00 00 00 00 30 4a b7 02 00 00 00 00 38 4a b7 02 00 00 00 00 70 7a c8 4e 66 7f 00 00
Individual Stats for: COMMAND_CLASS_MULTI_INSTANCE_ASSOCIATION (0x8e) --- Number of messages sent from this CommandClass. --- Number of messages received from this CommandClass.	 0 0
Individual Stats for: COMMAND_CLASS_VERSION (0x86) --- Number of messages sent from this CommandClass. --- Number of messages received from this CommandClass.	 0 0
Individual Stats for: COMMAND_CLASS_ASSOCIATION (0x85) --- Number of messages sent from this CommandClass. --- Number of messages received from this CommandClass.	 3 3
Individual Stats for: COMMAND_CLASS_POWERLEVEL (0x73) --- Number of messages sent from this CommandClass. --- Number of messages received from this CommandClass.	 1 1
Individual Stats for: COMMAND_CLASS_MANUFACTURER_SPECIFIC (0x72) --- Number of messages sent from this CommandClass. --- Number of messages received from this CommandClass.	 0 0
Individual Stats for: COMMAND_CLASS_CONFIGURATION (0x70) --- Number of messages sent from this CommandClass. --- Number of messages received from this CommandClass.	 19 19
Individual Stats for: COMMAND_CLASS_MULTI_CHANNEL_V2 (0x60) --- Number of messages sent from this CommandClass. --- Number of messages received from this CommandClass.	 2 2
Individual Stats for: COMMAND_CLASS_SWITCH_ALL (0x27) --- Number of messages sent from this CommandClass. --- Number of messages received from this CommandClass.	 1 1
Individual Stats for: COMMAND_CLASS_SWITCH_BINARY (0x25) --- Number of messages sent from this CommandClass. --- Number of messages received from this CommandClass.	 0 0
Individual Stats for: COMMAND_CLASS_BASIC (0x20) --- Number of messages sent from this CommandClass. --- Number of messages received from this CommandClass.	 1 1
Individual Stats for: COMMAND_CLASS_NO_OPERATION (0x0) --- Number of messages sent from this CommandClass. --- Number of messages received from this CommandClass.	 0 0

Device test messages

You can send test message to all active nodes, or just one. Enter node number, number of message and click button "Send test messages". Results are display in "Node statistic" section for both case.

```
Node 2 success test 1/3 in 19 ms.
Node 2 success test 2/3 in 44 ms.
Node 2 success last test 3 in 18 ms, all tests in 94 ms.
Node 5 success test 1/3 in 553 ms.
Node 5 success test 2/3 in 520 ms.
Node 5 success last test 3 in 451 ms, all tests in 1629 ms.
Node 7 success test 1/3 in 19 ms.
Node 7 success test 2/3 in 18 ms.
Node 7 success last test 3 in 19 ms, all tests in 1691 ms.
Node 8 success test 1/3 in 3182 ms.
Node 10 success test 1/3 in 23 ms.
Node 10 success test 2/3 in 19 ms.
Node 10 success last test 3 in 19 ms, all tests in 4947 ms.
Test Node 8 as recevied time out (10000 ms), 1/3 received.
```

CHAPTER 6

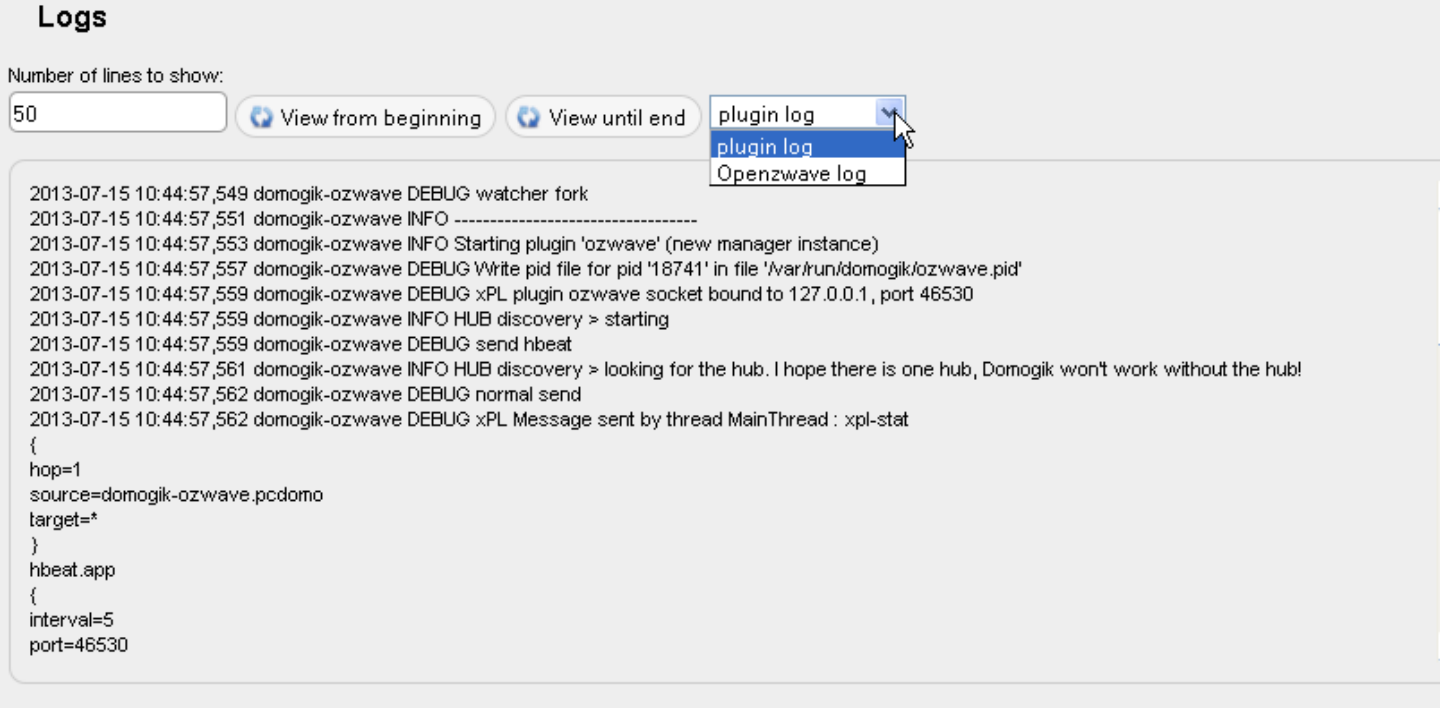
Support tools

Memory usage

- A click on button “refresh” show you how memory is use by plugin. It is an estimation.

Show log file informations

- You can display n lines from beginning or until end of ozwave plugin log file or openzwave lib C++ log when is activate in plugin parameter.
- Select type log, number of line(s) and click button.
- Negative value on number of lines shows full log.



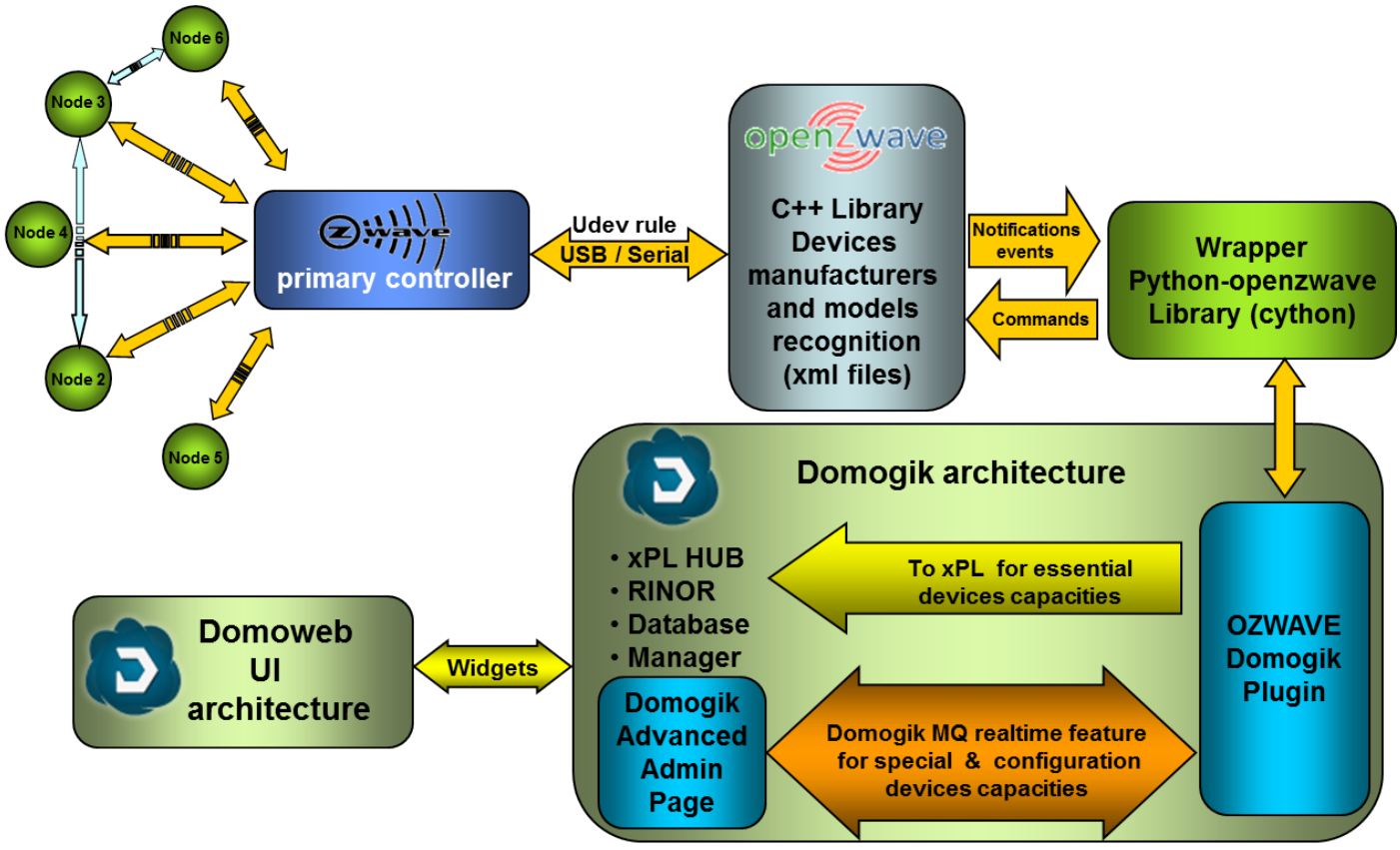
List of manufacturers and product compatibilities

- You can check all manufacturers and products reognized by openzwave library.

CHAPTER 7

Development informations

Detailed architecture



CHAPTER 8

Change log

0.5.4 : (13-05-2017)

- **Change log :**
  - Python-openzwave 0.4 compatibility tested.
  - Add **install_dependencies.sh -pip** option for python-openzwave >=0.4.0 install from pip.
  - Make python-openzwave 0.3.3 as default version used by install_dependencies.sh.
  - **Json Change :**
    - Add Danfos Popp Popp Radiator Thermostat.
    - Add Everspring AN180 Plug-in ON/OFF Module (Test Power management alarm as Trigger mod).
    - Add NorthQ NQ-9121 Gas Meter support.
    - Reload json class conversion file only within 60s.
  - Improve unreadable ozw value diagnostic.
  - Exclude existing sensor/command dmg device in detected device.
  - Improve user starting plugin informations in admin.
  - Convert zw cubic meter to m3 domogik unit.
  - Remove admin debug-pane.
  - Fix, dmgdevicelink case no product.
  - Fix product empty in dmgProducts.
  - Fix advanced admin plugin status host id.

0.5.3 : (27-12-2016)

- **Change log :**
  - **Json Change :**
    - Use DT_Motion
    - Rename some product references
    - Add products “AEOTEC ZW089 Recessed Door Sensor Gen5”, “QUBINO ZMNHCDx Flush Shutter”, “QUBINO ZMNHJD1 heating pilot wire”.
    - Add “pilot-wire-level” sensor/command
    - Add some device_types and fix somme typo.
    - Add NorthQ NQ-9121 Gas Meter Product and device type.
  - Handle CommandClass Alarm as string sensor.
  - Change pictures product to .png
  - Handle products domogik devices plugin associated to detected devices and known zwave modul.
  - Improved domogik device detection.
  - Improve Advanced Admin user information for node link to domogik device.
  - Add in Advanced Admin node xml detail from openzwave config displayed by xonomy js lib. TODO: Edit/save mode.
  - Fix Advanced Admin graph neighbors zoom, improve display.
  - Fix Advanced Admin display network menu on overlay.
  - Fix some Unicode.
  - Update doc.

0.5.2 : (03-08-2016)

- **Change log :**
  - **Json Change :**
    - Remove **Humidity** sensor, use only **Relative humidity**
    - Create general conversion **for_Binary_Inversion** and sensors binary inversion
    - Modify binary **Bright/Dim Increase/Decrease Up/down** button as a single command
    - Modify device_types associate
    - Add details configuration values openzwave options help
  - Handle **Bright/Dim Increase/Decrease Up/down** button
  - Update doc

0.5.1 : (27-06-2016)

- Compatibily library : OpenZwave >= 1.4.248, python_openzwave >= 0.3.0 (final)
- You must compile and install python_openzwave
- **Change log :**
  - install_dependencies.sh get python-openzwave archive lib from GitHub repository
  - Hide zwave network key in admin openzwave options.
  - Fix possible iterator issue on values iteration.
  - Add user notification on failed controller command.
  - Add node details info and zwave+ info.
  - Add user Domogik device detection refresh command.
  - Add user COMMAND_CLASS_CONFIGURATION refresh value command (global and individual).
  - Add AEON Z-Stick Gen5 udev rules
  - Handle Group Association instance.
  - Handle full hard reset with no resart needed.

0.5.0 : (11-05-2016)

- Compatibily library : OpenZwave >= 1.4, python_openzwave >= 0.3.0b8
- Compatibility with Domogik 0.5+ and non xpl devices
- Domogik devices must be recreate.(All device type are renamed)
- **Change log :**
  - Auto refresh device list (From MQ publish).
  - **info.json file, renamed device_type, sensors and commands.**
    - sensors and commands rule “type-detail” space replaced by “-”.
    - device_type rule : “ozwave.<”-”.join(sensor list)”__””-”.join(command list).
  - Domogik device addressing with networkID, node ID, instance in global parameters.
  - Sensors adressing with key “name”.
  - Commands adressing with key “key” of command parameters.
  - Add linked label, definition in lib/linkedlabels.json
  - Add dynamic command_class conversion, definition in lib/cmd_class_conversion.json
  - Auto load domogik openzwave labels availaible from info.json.
  - Add Detected domogik devices.
  - Add graph neibhbors whitn new possitionning algorithm and dynamic update.
  - Handled non ASCII exception from python_openzwave 0.3.0b8.
  - Add install dependencies script.
  - Add refresh admin domogik device button.

  - Fix ManagerMonitorNodes stop issue.
  - Improve log.
  - Doc update.

0.4.0b1 : (12-10-2015) Compatibily library : OpenZwave >= 1.3.401, 0.3.0 (b6) <= python_openzwave >= 0.3.0 (b4)

- Target : python-openzwave, domogik (0.4.1)
- Update, compile and install python_openzwave.
- New version for domogik 0.4.1

0.4.0a1 : (30-03-2014) Compatibily library : OpenZwave >= 1.0.711 (/branches/2013-11-13_release_testing), python_openzwave >= 0.2.5 (rev >= 3bef0f1cb27f)

- Target : python-openzwave, domogik (0.4a1)
- Initial version for domogik 0.4 - Source plugin 0.2c4 for domogik 0.3
- Not usuable, developpement version.

Historical for domogik version 0.3.x

- **0.2c4** [(31-01-2014) Compatibily library][OpenZwave >= 1.0.711 (/branches/2013-11-13_release_testing), python_openzwave >= 0.2.5 (rev >= 3bef0f1cb27f)]
  - Target : python-openzwave, domogik (0.2c4), domoweb (0.2c3)
  - An insert_data or a reinstall is necessary.
  - **Update, compile and install python_openzwave.**
    - checkout openzwave branch : goto python-openzwave/openzwave,
    - do command: svn checkout http://open-zwave.googlecode.com/svn/branches/2013-11-13_release_testing
  - **Change log :**
    - Add heal node/network functions.
    - Add command_class_sensor_alarm.
    - Add auto battery level request at wakeup node.
    - Update doc
- **0.2c3** [(06-01-2014) Compatibily library][OpenZwave >= 1.0.706, python_openzwave >= 0.2.5 (rev >= 75d9b6a8dd77), Compatibilty with ws4py last update >= 0.3.3]
  - Target : OpenZwave, python-openzwave, ws4py, domogik (0.2c3), domoweb (0.2c2)
  - Update ws4py with pip or easy_install or update source and compile it.

  - Update, compile and install python_openzwave.
  - **Change log :**
    * GetPollIntensity add in openzwave lib.
    * Add log debug information for wsServer.
    * Fix auto-startup failure (rest no response).
    * Fix automatique COMMAND_CLASS_SENSOR_BINARY type set for xpl (recreate devices and widget for changes).
    * Fix issue in monitornodes xpl-report if no conforme xpl_msg.
    * Fix wsClient error if ack = undefined (domoweb).
    * Force default name for controler device if not defined (CtrlMustBeCreate.1.1).
    * Update doc

- **0.2c2** [(30-10-2013) Compatibily library][The same, Warning in case of ws4py install or update use >= 0.3.0-beta rev eec3a7dcb33b322eac598f5125425e62c0050969, see instructions in dependencies page.]
  - Target : domogik (0.2c2), domoweb (0.2c1)
  - An insert_data or a reinstall is necessary.
  - Knows issue : At auto start plugin fail sometimes, stop it or kill process ($ps -ef | grep ozwave) and restart it.
  - **Change log :**
    * Add individual monitoring node(s) in log file for debugging and helper develop device compatibilities.
    * Add removing ghost node from groups capability.
    * Add Dimmer multi-controls (FGD211) and bright/dim commands.
    * Add Thermostat setpoint (Danfos living-connect)
    * Add NotifyTransactions option (You must probably modify openzwave lib to fix issues failling plugin start, see doc section options .)
    * Fix wsServer fail connection some time.
    * Fix memory change during websocket sending Broadcast and Ack Message.
    * Fix no display “Support tools” page with some browsers versions.
    * Fix select last controleur action in reopen dialog box.
    * Graph neighbors code improved.
    * Update doc

- **0.2c1** [(15-09-2013) Compatibily library][The same, Warning in case of ws4py install or update use >= 0.3.0-beta rev eec3a7dcb33b322eac598f5125425e62c0050969, see instructions in dependencies page.]
  - Target : domogik.
  - **Change log :**
    * Retrieve domogik device zwave ctrl address from rest.
    * Update doc

- **0.2b5** [(29-07-2013) Compatibily library][OpenZwave >= 1.0.663, python_openzwave >= 0.2.5 (rev >= 18832df1dd95).]
  - Target : python_openzwave, domogik and domoweb.
  - Update, compile and install python_openzwave.
  - An insert_data or a reinstall is necessary.
  - **Change log :**
    * Fix some report controller action.
    * Fix accent on node name and location.
    * Fix display update name and location name just after sended.
    * Fix some actions bug.
    * Fix some Exception error.
    * Forced unit conversion F -> °C for temperature device in F.
    * Double xPL message for switch multi-level. (for testing-not sure it's a good idea !)
    * UI dialog node associations improvement.
    * Add device motion binary multi sensor 4-in-1.
    * Add polling command_class. Some limitations : there is an issue with getPollIntensity openzwave lib so for moment function is deactivated and value 1 is always received. Some command class like COMMAND_CLASS_POWERLEVEL seem to enable but the polling is not in effect.
    * Add timer reporting controller status on widget (every 60s) and report status "started plugin, started, init, lock, no-ctrl, ok, stop".
    * Add list of recognized manufacturers and products by openzwave in "support tools" tab.
    * Add force refresh node.
    * Add openzwave log report in "support tools" tab.
    * Add Battery status in tab nodes.
    * Update doc.
- **0.2b4** [(05-28-2013) Compatibily library][the same.]
  - Target : domogik and domoweb
  - **Change log :**
    * Fixe bug plugin starting with package installation
    * Add Domoweb version. (first 0.2.b4)
    * Add support tools, memory usage and log report to UI.
    * Change websocket server private plugin to generic usage.
    * Add load xml open-zwave usage, don't finish coding.
- **0.2b3** [Compatibily library][python_openzwave >= 0.2.5 (rev][b434c50b795b), tailer >=0.2.1]
  - Target : python_openzwave, domogik and domoweb
  - Update, compile and install python_openzwave
  - Install tailer : "sudo pip install tailer" or "sudo easy_install tailer"

  - **Change log :**
    - Add test network and node
    - Add id message req-ack
- **0.2b2** [Compatibily library][the same.]
  - Target : domogik and domoweb
  - An insert_data or a reinstall is necessary
  - **Change log :**
    - Check user directory and config directory acces
    - Realtime improvements for graph neighbors
    - Add Start/Stop driver function
    - Add zwave device switch with power meter (Everspring (C.T.) - AN158 full handling)
    - Fixe ON/OFF sensor return status
- **0.2b1** [Compatibily library][OpenZwave >= 1.0.645, python_openzwave >= 0.2.5, ws4py >= 0.3.0-beta]
  - Target : domogik and domoweb
  - Install new dependency ws4py 0.3.0-beta
  - An insert_data or a reinstall is necessary
  - Enter new wsportserver key (Plugin configuration) and save the config (necessary to restart plugin if is started)
- **0.1b8 :**
  - An insert_data or a reinstall is necessary.
  - Created primary controller device for domogik and traced back to the state it to domogik.

Do an insert data

In your domogik directory with user domogik :

```
$ src/tools/packages/insert_data.py src/share/domogik/plugins/ozwave.json
```

CHAPTER 9

Advanced - Dependencies installation

Install the tailer library for Python

- tailer 0.3 library for Python

Install **tailer** :

```
$ sudo pip install tailer
```

Install python-openzwave

Purpose

This not a plugin, but an external library for zwave plugin, is based on python-openzwave software.

python-openzawe and openzawe are in high development, by two different teams, so installing it can be sometimes not so easy. So we propose you different methods to install it.

For the operation of ozwave domogik plugin there is only need part python-openzwave Lib.

Of course you can install part python-openzwave API

Get information from bibi21000 home site form more python-openzwave details.

All information are extracted from the git repository https://github.com/OpenZWave/python-openzwave

Installing python-openzwave from archive

This is the simplest (and the fastest) way to install python- openzwave. It cames with openzwave source files and is already cythonized.

This is surely the best solution to install python-openzwave on a raspberry pi.

Get archive of python-openzwave

You are now ready to download sources of python-openzwave here :

http://bibi21000.no-ip.biz/python-openzwave/

This archive contains sources of python-openzwave and openzwave.

```
tar xvzf python-openzwave-X.Y.Z.tar.gz
```

This command will extract all the needed sources. And change to the right directory.

```
cd python-openzwave-X.Y.Z
```

Install the needed tools

You must install git and other tools to get sources of python- openzwave and openzwave and build them. Look at the documentation of your Linux distribution to do that.

On a debian like distribution :

```
sudo make deps
```

Build process

Now, you can compile sources :

```
make build
```

If you have already built python-openzwave or the build failed you can use the clean option :

```
sudo make clean
make build
```

Do not use root to build python-openzwave as it will surely fails. Please use a “normal user”.

Installation

You can now install the packages using the following command will.

```
sudo make install
```

The installation script create a list of installed files. So you can remove python-openzwave using the following command :

```
sudo make uninstall
```

If it fails

Simply remove the python-openzwave-x.y.z directory and extract it again.

Installing python-openzwave from repository

Install the needed tools

You must install git and make to retrieve sources of python-openzwave and openzwave.

On a debian like distribution :

```
sudo apt-get install -y git make
```

Get sources of python-openzwave

You are now ready to download sources of python-openzwave :

```
git clone https://github.com/OpenZWave/python-openzwave
```

The previous command will create a copy of the official repository on your computer in a directory called python-openzwave.

Install dependencies

You need some tools (a c++ compiler, headers dir python, ...) to build python-openzwave and openzwave library.

On a debian like distribution :

```
sudo make repo-deps
```

For non-debian (fedora, ...), you can retrieve the packages needed in the Makefile.

Update and build process

Go to the previously created directory

```
cd python-openzwave
```

The following command will update your local repository to the last release of python-openzwave and openzwave.

```
make update
```

When update process is done, you can compile sources

```
make build
```

Or if you have already build python-openzwave in a previous installation, you can use the clean target to remove old builds.

```
sudo make clean
```

Do not use root to build python-openzwave as it will surely fails. Please use a “normal user”.

Installation

You can now ready to install the eggs using the following command :

```
sudo make install
```

You can also remove python-openzwave using :

```
sudo make uninstall
```

Running tests

You can launch the regression tests using :

```
make tests
```

Keep in mind that the tests will "play" with your nodes : switching on and off, dimming, adding and removing scenes, ...

Static vs dynamic (or shared)

The openzwave (c++) lib needs to run as a singleton : it means that it MUST have only one instance of the manager running on your computer.

There is 2 ways of linking libraries with a program :

static : includes a copy of the library in your binary program. This means

that your program has its own instance of the library. This the way the install.sh runs. So you CAN'T have another program (like the control-panel) running when using the python-openzwave library

dynamic or shared : includes a link to the library in your binary program.

This means that your program share the library with other programs. In this case, the instance is owned directly by the library. This the way the debian package works. So you CAN have another program running when using the python-openzwave library. Of course, this program MUST use the shared library.

Creating the zwave device controller

We need to create an udev rule in order to create the device **/dev/zwave** - check your device It's suppose your zwave controller is at **/dev/ttyUSB0**

```
$ udevadm info --name=/dev/ttyUSB0 --attribute-walk
```

- locate your idVendor and idProdroduct

In **/etc/udev/rules.d**, create a file zwave.rules, and write the following rule

Example, for aeon stick

```
| SUBSYSTEMS=="usb", ATTRS{idVendor}=="10c4", ATTRS{idProduct}=="ea60",
                SYMLINK+="zwave", MODE="0666"
```

Other zwave tool

Migrating from python-openzwave 0.2.X to 0.3.0

I need to update source tree of python-openzwave and modules's names because of a bug in setuptools : https://bitbucket.org/pypa/setuptools/issue/230/develop-mode-does-not-respect-src . Sorry for that.

Update your sources:

```
git pull
```

Before building python-openzwave 0.3.0, you must uninstall the old version :

```
sudo make uninstall
```

About cython : I've made many tests using cython installed via pip : (0.20, 0.21 and 0.22). Compilation is ok but a segfault appears when launching the tests. Please remove it.

```
sudo pip uninstall Cython
```

You also need to make some minor updates in you code, look at CHANGELOG

If you have problems, please submit an issue with :

- cython -V
- the content of the directory /usr/local/lib/python2.7/dist-packages/ (for python2.7)
- the content of /usr/local/lib/python2.7/dist-packages/easy-install.pth (for python 2.7)

Ubuntu 64bits ... and the others

If you're using Ubuntu 64 bits (and mayde others) and keep your distribution up to date, you certainly have the segfault problem.

It appears with the last update of python :

```
$ python
Python 2.7.6 (default, Jun 22 2015, 17:58:13)
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
```

I've open a discussion on cython-users here : https://groups.google.com/forum/#!topic/cython-users/mRsviGuCFOk

The only way I found to avoid this is to rebuild and reinstall the old release of python :

```
wget https://launchpad.net/ubuntu/+archive/primary/+files/python2.7_2.7.6-8.dsc␣
↪https://launchpad.net/ubuntu/+archive/primary/+files/python2.7_2.7.6.orig.tar.gz␣
↪https://launchpad.net/ubuntu/+archive/primary/+files/python2.7_2.7.6-8.diff.gz

dpkg-source -x python2.7_2.7.6-8.dsc

sudo apt-get build-dep python2.7

cd python2.7-2.7.6

dpkg-buildpackage
```

Wait, wait and await again :)

```
cd ..

sudo dpkg -i *.deb
```

To prevent future updates of python, you could mark its packages. For example, if you use apt to update your distribution, use the following command :

```
sudo apt-mark hold idle-python2.7 libpython2.7-minimal python2.7-dbg python2.7-
↪minimal libpython2.7 libpython2.7-stdlib python2.7-dev libpython2.7-dbg  libpython2.
↪7-testsuite python2.7-doc libpython2.7-dev python2.7 python2.7-examples
```

Some users have reported that building python-openzwave using the archive (INSTALL_ARCH) can also do the trick. Let me know if it works for you.

Openzwave control-panel

In order to identify your network and collect the NodeID of your devices, you can use the openzwave-control-panel

Developer resources

For developing you can access to python-openzwave dev, instructions here :

http://bibi21000.gallet.info/index.php/en/component/sphinxdoc/documentation/4-python-openzwave-lib.html

http://bibi21000.gallet.info/index.php/en/component/sphinxdoc/documentation/3-python-openzwave-api.html

CHAPTER 10

Advanced - Usage of udev rules

This page is dedicated to the users who want to create their own **udev rules** file!

Please keep in mind that you should use, if possible, one of the **udev rules** files delivered with this plugin!

Create the udev rule for controller

Gather information about your device controller (USB)

- Example using Aeon Stick2 on USB port. For ohter model it's should different.
- Use **lsusb** command for listing of USB devices, check before and after plug your USB controller.

```
$ lsusb
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 004 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 005 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 006 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 007 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 008 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 009 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 005 Device 002: ID 0040:073d
Bus 004 Device 002: ID 10c4:ea60 Cygnal Integrated Products, Inc. CP210x Composite␣
↪Device
Bus 004 Device 006: ID 0403:6001 Future Technology Devices International, Ltd FT232␣
↪USB-Serial (UART) IC
Bus 007 Device 002: ID 0b05:179c ASUSTek Computer, Inc.
```

- Use **ls /dev/ttyUSB*** to check your num USB, check before and after plug your USB controller.
- Before

```
$ ls /dev/ttyUSB*
/dev/ttyUSB2
```

- After USB plug

```
$ ls /dev/ttyUSB*
/dev/ttyUSB0  /dev/ttyUSB2
```

- Use **udevadm** command to gather information about your device :

```
$ udevadm info  -a -n /dev/ttyUSB0

Udevadm info starts with the device specified by the devpath and then
walks up the chain of parent devices. It prints for every device
found, all possible attributes in the udev rules key format.
A rule to match, can be composed by the attributes of the device
and the attributes from one single parent device.

  looking at device '/devices/pci0000:00/0000:00:12.0/usb4/4-3/4-3:1.0/ttyUSB0/tty/
↪ttyUSB0':
    KERNEL=="ttyUSB0"
    SUBSYSTEM=="tty"
    DRIVER==""

  looking at parent device '/devices/pci0000:00/0000:00:12.0/usb4/4-3/4-3:1.0/ttyUSB0
↪':
    KERNELS=="ttyUSB0"
    SUBSYSTEMS=="usb-serial"
    DRIVERS=="cp210x"
    ATTRS{port_number}=="0"

  looking at parent device '/devices/pci0000:00/0000:00:12.0/usb4/4-3/4-3:1.0':
    KERNELS=="4-3:1.0"
    SUBSYSTEMS=="usb"
    DRIVERS=="cp210x"
    ATTRS{bInterfaceNumber}=="00"
    ATTRS{bAlternateSetting}==" 0"
    ATTRS{bNumEndpoints}=="02"
    ATTRS{bInterfaceClass}=="ff"
    ATTRS{bInterfaceSubClass}=="00"
    ATTRS{bInterfaceProtocol}=="00"
    ATTRS{supports_autosuspend}=="1"
    ATTRS{interface}=="CP2102 USB to UART Bridge Controller"

  looking at parent device '/devices/pci0000:00/0000:00:12.0/usb4/4-3':
    KERNELS=="4-3"
    SUBSYSTEMS=="usb"
    DRIVERS=="usb"
    ATTRS{configuration}==""
    ATTRS{bNumInterfaces}==" 1"
    ATTRS{bConfigurationValue}=="1"
    ATTRS{bmAttributes}=="80"
    ATTRS{bMaxPower}=="100mA"
    ATTRS{urbnum}=="10835"
    ATTRS{idVendor}=="10c4"
    ATTRS{idProduct}=="ea60"
    ATTRS{bcdDevice}=="0100"
    ATTRS{bDeviceClass}=="00"
```

```
    ATTRS{bDeviceSubClass}=="00"
    ATTRS{bDeviceProtocol}=="00"
    ATTRS{bNumConfigurations}=="1"
    ATTRS{bMaxPacketSize0}=="64"
    ATTRS{speed}=="12"
    ATTRS{busnum}=="4"
    ATTRS{devnum}=="2"
    ATTRS{devpath}=="3"
    ATTRS{version}==" 1.10"
    ATTRS{maxchild}=="0"
    ATTRS{quirks}=="0x0"
    ATTRS{avoid_reset_quirk}=="0"
    ATTRS{authorized}=="1"
    ATTRS{manufacturer}=="Silicon Labs"
    ATTRS{product}=="CP2102 USB to UART Bridge Controller"
    ATTRS{serial}=="0001"

  looking at parent device '/devices/pci0000:00/0000:00:12.0/usb4':
    KERNELS=="usb4"
    SUBSYSTEMS=="usb"
    DRIVERS=="usb"
    ATTRS{configuration}==""
    ATTRS{bNumInterfaces}==" 1"
    ATTRS{bConfigurationValue}=="1"
    ATTRS{bmAttributes}=="e0"
    ATTRS{bMaxPower}=="  0mA"
    ATTRS{urbnum}=="134"
    ATTRS{idVendor}=="1d6b"
    ATTRS{idProduct}=="0001"
    ATTRS{bcdDevice}=="0300"
    ATTRS{bDeviceClass}=="09"
    ATTRS{bDeviceSubClass}=="00"
    ATTRS{bDeviceProtocol}=="00"
    ATTRS{bNumConfigurations}=="1"
    ATTRS{bMaxPacketSize0}=="64"
    ATTRS{speed}=="12"
    ATTRS{busnum}=="4"
    ATTRS{devnum}=="1"
    ATTRS{devpath}=="0"
    ATTRS{version}==" 1.10"
    ATTRS{maxchild}=="5"
    ATTRS{quirks}=="0x0"
    ATTRS{avoid_reset_quirk}=="0"
    ATTRS{authorized}=="1"
    ATTRS{manufacturer}=="Linux 3.0.0-24-generic ohci_hcd"
    ATTRS{product}=="OHCI Host Controller"
    ATTRS{serial}=="0000:00:12.0"
    ATTRS{authorized_default}=="1"

  looking at parent device '/devices/pci0000:00/0000:00:12.0':
    KERNELS=="0000:00:12.0"
    SUBSYSTEMS=="pci"
    DRIVERS=="ohci_hcd"
    ATTRS{vendor}=="0x1002"
    ATTRS{device}=="0x4397"
    ATTRS{subsystem_vendor}=="0x1043"
    ATTRS{subsystem_device}=="0x8496"
    ATTRS{class}=="0x0c0310"
```

```
    ATTRS{irq}=="18"
    ATTRS{local_cpus}=="00000000,00000000,00000000,00000000,00000000,00000000,
↪00000000,00000003"
    ATTRS{local_cpulist}=="0-1"
    ATTRS{numa_node}=="-1"
    ATTRS{dma_mask_bits}=="32"
    ATTRS{consistent_dma_mask_bits}=="32"
    ATTRS{broken_parity_status}=="0"
    ATTRS{msi_bus}==""

  looking at parent device '/devices/pci0000:00':
    KERNELS=="pci0000:00"
    SUBSYSTEMS==""
    DRIVERS==""
```

Those information will be useful to determinate for sure that this device is your Zwave controller. We will use several information, flagged above as **DRIVERS ATTRS{manufacturer}** and **ATTRS{product}** With that, we will be sure that we'll be talking to our controller. You can chose others attributs.

Create the rule

- Create a new file, in folder **etc/udev/rules.d** Let's call it **98-usbcp210x.rules**
- Enter those information in the file :

```
# for z-Stick serie 2 to domogik /dev/zwave
DRIVERS=="usb", ATTRS{manufacturer}=="Silicon Labs", ATTRS{product}=="CP2102 USB to␣
↪UART Bridge Controller", SYMLINK+="zwave", MODE="0666"
```

The **DRIVERS ATTRS{manufacturer} ATTRS{product}** values must be coherent with what you have found above. * Ask udev to rediscover your device :

```
# udevadm test $(udevadm info --query path --name ttyUSB0)
```

- Your device should now be re-discovered, let's confirm it :

```
$ ls -l /dev/zwave
lrwxrwxrwx 1 root root 7 2012-08-27 00:46 /dev/zwave -> ttyUSB0
```