Include S2 Devices

You can validate the inclusion of S2 devices with our API tool:

- Login at https://apitool.ezlo.com/dashboard with your Ezlo/Vera/MiOS username and password.
- Select the controller to which you want to add devices then click ‘Connect’.
- Select ‘custom’ in the ‘Commands’ drop-down menu.
- Paste the following call into the field below the drop-down:

```
{
 "method": "hub.extensions.plugin.run",
 "id": "random1",
 "params": {
  "script": "HUB:zwave/scripts/start_include"
 }
}
```

- Click ‘Send’ to put the hub in inclusion mode.

The hub will reply using the same id:

```
{
 "method": "hub.extensions.plugin.run",
 "result": {},
 "error": null,
 "id": "random1",
 "sender": {
  "conn_id": "e09ff25e-8d90-4c0e-bf8f-2d325181c751",
  "type": "ui"
 }
}
```

and will start the broadcast indicating that the process was invoked but not started yet (the **conn_id** value is a random string assigned to the socket client):

```
{
 "id": "ui_broadcast",
 "msg_subclass": "hub.extensions.plugin.ui_broadcast",
 "result": {
  "event": "include_invoked",
  "plugin": "zwave"
 }
}
```


- Put the device in inclusion mode. The hub will send a message stating the inclusion has started:


```
{
 "id": "ui_broadcast",
 "msg_subclass": "hub.extensions.plugin.ui_broadcast",
 "result": {
  "event": "include_started",
  "plugin": "zwave"
 }
}
```


- The hub will broadcast the security modes allowed by the device:


```
{
 "id": "ui_broadcast",
 "msg_subclass": "hub.extensions.plugin.ui_broadcast",
 "result": {
  "event": "s2_select_authentication_modes",
  "plugin": "zwave",
  "modes": [
   "accessControl",
   "s0"
  ]
 }
}
```


Where **modes** is an array of available authentication modes, the possible values are:

➔ accessControl
➔ authenticated
➔ unauthenticated
➔ s0

➔ clientSide

- Send a message with the desired modes, this must be done within 240 seconds. However, some devices have a lower timeout:

```
{
 "method": "hub.extensions.plugin.run",
 "id": "random2",
 "params": {
  "script": "HUB:zwave/scripts/set_authentication_mode",
  "scriptParams": {
   "modes": [
    "unauthenticated",
    "authenticated",
    "s0"
   ]
  }
 }
}
```

The hub will reply with the same id of the call:

```
{
 "method": "hub.extensions.plugin.run",
 "result": {},
 "error": null,
 "id": "random2",
 "sender": {
  "conn_id": "e09ff25e-8d90-4c0e-bf8f-2d325181c751",
  "type": "ui"
 }
}
```

And will subsequently broadcast the key value - an array of 7 items that represent the device-specific key (public part):

```
{
 "id": "ui_broadcast",
 "msg_subclass": "hub.extensions.plugin.ui_broadcast",
 "result": {
  "event": "s2_request_device_specific_key",
  "plugin": "zwave",
  "key": [
   "12345",
   "67890",
   "12345",
   "67890",
   "12345",
   "67890",
   "12345"
  ]
 }
}
```


- Send a message with the private part of the device-specific key (1 number). This must be done within 240 seconds, however, some devices have a lower timeout:


```
{
 "method": "hub.extensions.plugin.run",
 "id": "random3",
 "params": {
  "script": "HUB:zwave/scripts/set_device_specific_key",
  "scriptParams": {
   "key": 99999
  }
 }
}
```


The hub will reply with the same id:

```
{
 "method": "hub.extensions.plugin.run",
 "result": {},
 "error": null,
 "id": "random3",
 "sender": {
  "conn_id": "e09ff25e-8d90-4c0e-bf8f-2d325181c751",
  "type": "ui"
 }
}
```


After that, several broadcast messages can be seen depending on the items that the device implements (similar to those seen when adding a Z-Wave device). The hub will indicate the inclusion ended successfully once the configuration is ready :


```
{
 "id": "ui_broadcast",
 "msg_subclass": "hub.extensions.plugin.ui_broadcast",
 "result": {
  "event": "include_finished",
  "plugin": "zwave"
 }
}
```


It is possible to see some additional broadcast messages after the inclusion. These are triggered when the device reports its current status to the hub. These messages will mostly have **"id"**: **"ui_broadcast"** , and **"msg_subclass": "hub.item.updated"** in them. For example:

```
{
 "id": "ui_broadcast",
 "msg_subclass": "hub.item.updated",
 "result": {
  "_id": "user_codes_scan_progressEE8A55D2",
  "deviceId": "ZA08FBACB",
  "deviceName": "Danalock V3",
  "deviceCategory": "door_lock",
  "deviceSubcategory": "",
  "serviceNotification": false,
  "roomName": "",
  "userNotification": false,
  "notifications": [],
  "name": "user_codes_scan_progress",
  "valueType": "int",
  "value": 75,
  "minValue": 0,
  "maxValue": 100,
  "syncNotification": false
 }
}
```


Once added, the S2 device should appear when you select "hub.devices.list" in the ‘Commands’ drop-down menu.

Change mode and setpoint of a thermostat

Get the id of items to control modes and setpoint


```
{
 "method": "hub.items.list",
 "id": "1604354274921",
 "params": {}
}
```


Take note of the IDs for mode changes, setpoint changes. Also, the modes supported.

```
{
 "_id": "73C1C9D8",
 "deviceId": "Z7A391A25",
 "hasGetter": true,
 "hasSetter": true,
 "name": "thermostat_mode",
 "show": true,
 "valueType": "token",
 "value": "cool",
 "enum": [ "off", "heat", "cool", "auto", "aux" ]
}

{
 "_id": "72214C24",
 "deviceId": "Z7A391A25",
 "hasGetter": true,
 "hasSetter": true,
 "name": "thermostat_setpoint",
 "show": true,
 "valueType": "temperature",
 "value": 61,
 "scale": "fahrenheit",
 "minValue": 0,
 "maxValue": 212
}
```


Set the desired mode using the item value set call. Pass as a parameter one of the modes supported:

```
{
 "method": "hub.item.value.set",
 "id": "1604353070355",
 "params": {
  "_id": "73C1C9D8",
  "value": "heat"
 }
}
```

You can also use the item value set call to change the setpoint:

```
{
 "method": "hub.item.value.set",
 "id": "1603720927917",
 "params": {
  "_id": "72214C24",
  "value": 80,
  "scale": "fahrenheit"
 }
}
```