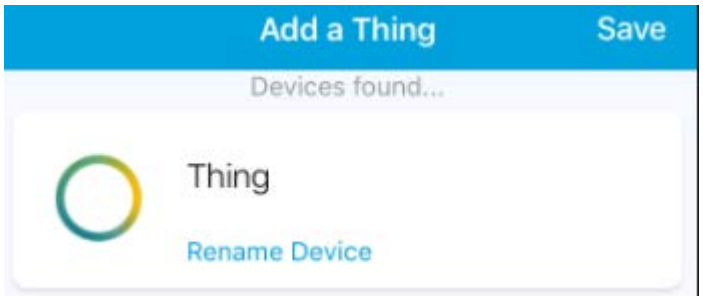
Add a Device Handler for Your SYLVANIA SMART+ ZigBee Smart Plug

This guide goes through how to add SYLVANIA SMART+ ZigBee Smart Plug.

Is Your Device Supported?

If you try to pair your SMART+ device to SmartThings, you may encounter the following display:



What this means is that SmartThings recognizes your ZigBee SMART+ plug, but doesn't know how to categorize it and what functionality to give it so you can control your light properly within the app. If you are experiencing this when trying to pair a SYLVANIA SMART+ product, then proceed with the following steps below to properly pair your products to your SmartThings set up.

Remove the Light from Your Set Up

If you attempt to pair your smart light and receive the above message, you must first remove the device from your SmartThings set up before proceeding.

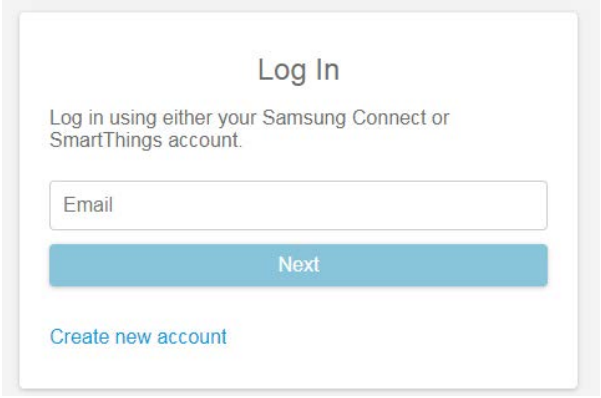
To do this, in the app, navigate to My Home > Things > and tap on the device you just paired. From there, go to "Settings" in the upper right hand corner and select "Remove Device."

Adding the Code to the SmartThings IDE

The SmartThings IDE is a place where you can manage your SmartThings locations, hubs, and devices on your desktop. It is here you will need to copy and paste a piece of code in order to pair your SMART+ products not fully supported.

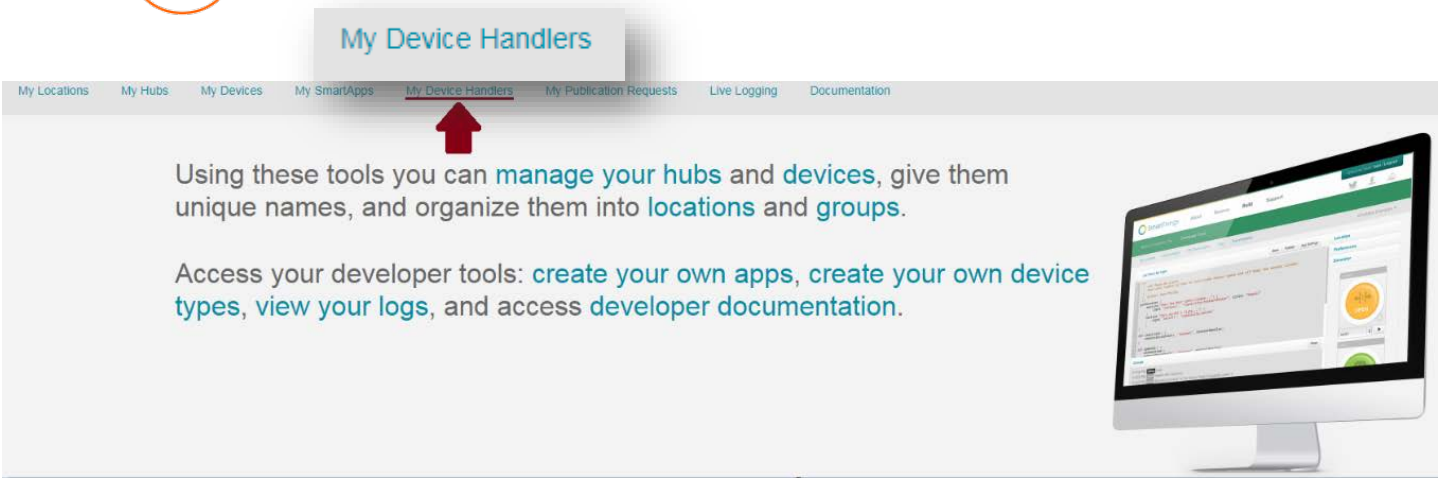
1

Navigate to https://graph.api.smartthings.com/ and login with your SmartThings account. This will be the same account information you use to log into your SmartThings mobile app.



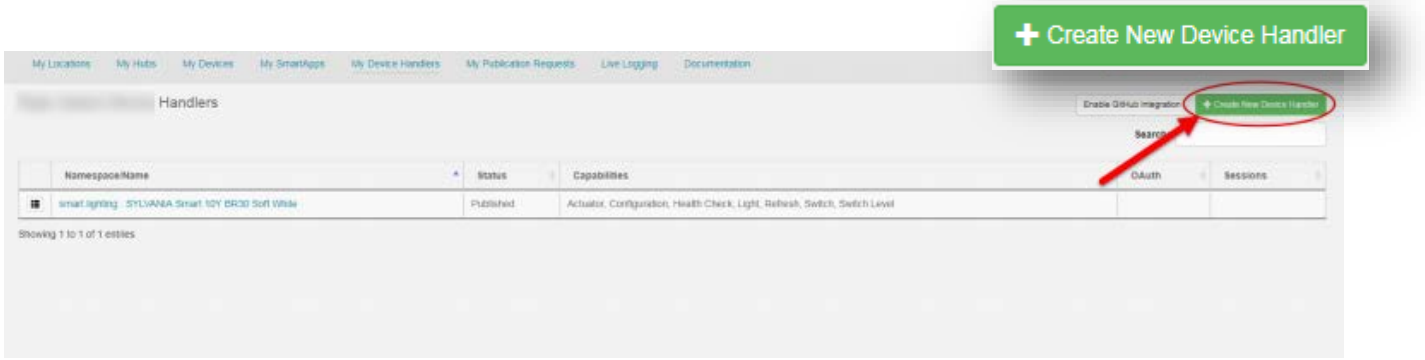
2

Once logged in, navigate to **My Device Handlers** at the top of the screen:



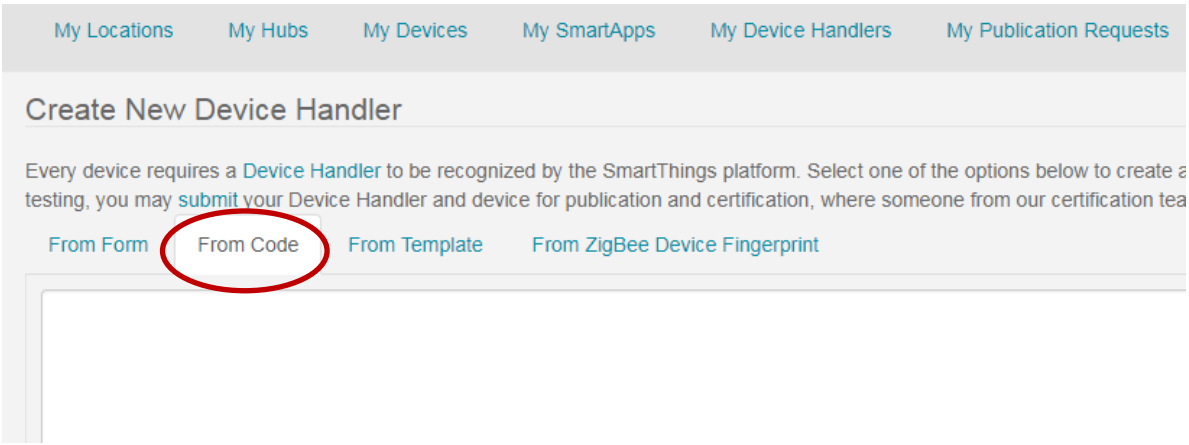
3

Click **Create New Device Handler** in the top right hand size of the screen.



LEDVANCE

4 Select the **From Code** tab.



5 Copy and paste the following code into the box below:

```
metadata {
	definition (name: "SYLVANIA Smart Plug", namespace: "ledvanceDH", author:
"Ledvance") {
		capability "Actuator"
		capability "Switch"
		capability "Power Meter"
		capability "Configuration"
		capability "Refresh"
		capability "Sensor"
		capability "Health Check"

		fingerprint profileId: "C05E", inClusters:
"1000,0000,0003,0004,0005,0006,0B04,FC0F", outClusters: "0019", manufacturer: "OSRAM",
model: "Plug 01", deviceJoinName: "SYLVANIA Smart Plug"
		fingerprint profileId: "0104", inClusters:
"0000,0003,0004,0005,0006,0B05,FC01,FC08", outClusters: "0003,0019", manufacturer:
"LEDVACE", model: "PLUG", deviceJoinName: "SYLVANIA Smart Plug"
	}

	// simulator metadata
	simulator {
		// status messages
		status "on": "on/off: 1"
		status "off": "on/off: 0"

		// reply messages
		reply "zcl on-off on": "on/off: 1"
		reply "zcl on-off off": "on/off: 0"
	}

	preferences {
		section {
			image(name: 'educationalcontent', multiple: true, images: [
				"http://cdn.device-
gse.smartthings.com/Outlet/US/OutletUS1.jpg",
				"http://cdn.device-
gse.smartthings.com/Outlet/US/OutletUS2.jpg"
				])
		}
	}
```

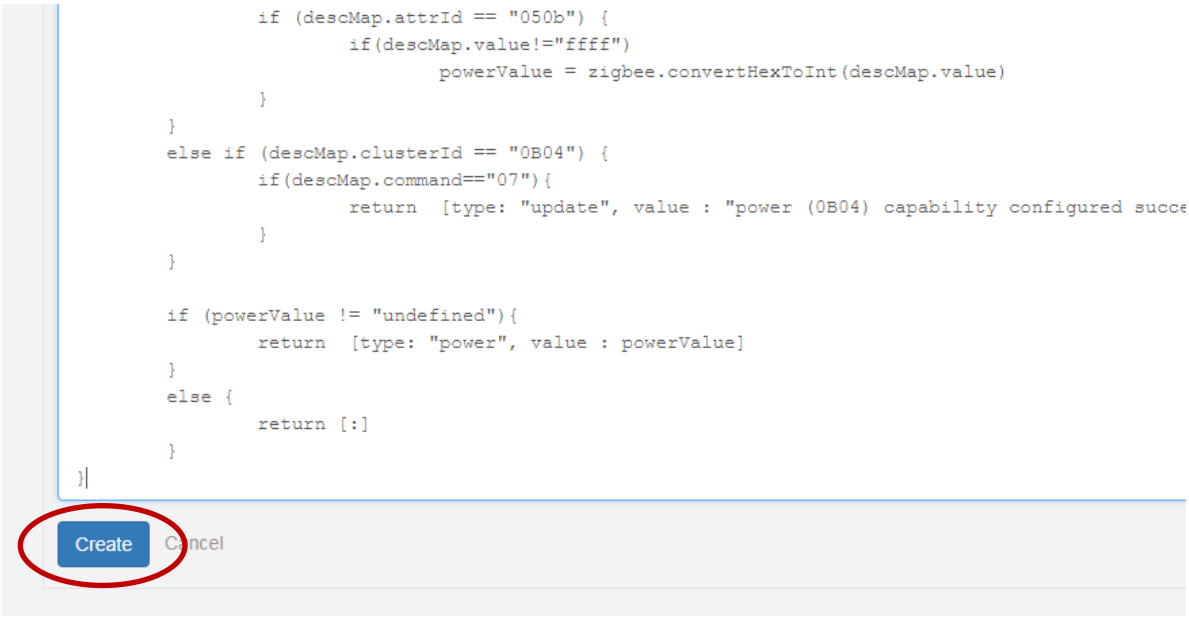
LEDVANCE

```
	// UI tile definitions
	tiles(scale: 2) {
		multiAttributeTile(name:"switch", type: "lighting", width: 6, height: 4,
canChangeIcon: true){
			tileAttribute ("device.switch", key: "PRIMARY_CONTROL") {
				attributeState "on", label: 'On', action: "switch.off",
icon: "st.Appliances.appliances17", backgroundColor: "#79b821", nextState: "turningOff"
				attributeState "off", label: 'Off', action: "switch.on",
icon: "st.Appliances.appliances17", backgroundColor: "#565C51", nextState: "turningOn"
				attributeState "turningOn", label: 'Turning On', action:
"switch.off", icon: "st.Appliances.appliances17", backgroundColor: "#60903A", nextState:
"turningOff"
				attributeState "turningOff", label: 'Turning Off', action:
"switch.on", icon: "st.Appliances.appliances17", backgroundColor: "#CACACA", nextState:
"turningOn"
			}
			tileAttribute ("power", key: "SECONDARY_CONTROL") {
				attributeState "power", label:'${currentValue} W'
			}
		}

		standardTile("refresh", "device.power", inactiveLabel: false, decoration:
"flat", width: 2, height: 2) {
			state "default", label:'', action:"refresh.refresh",
icon:"st.secondary.refresh"
		}

		main "switch"
		details(["switch","refresh"])
	}
}

// Parse incoming device messages to generate events
def parse(String description) {
	log.debug "description is $description"

	def finalResult = zigbee.getKnownDescription(description)
	def event = [:]

	//TODO: Remove this after getKnownDescription can parse it automatically
	if (!finalResult && description!="updated")
		finalResult =
getPowerDescription(zigbee.parseDescriptionAsMap(description))

	if (finalResult) {
		log.info "final result = $finalResult"
		if (finalResult.type == "update") {
			log.info "$device updates: ${finalResult.value}"
			event = null
		}
		else if (finalResult.type == "power") {
			def powerValue = (finalResult.value as Integer)/10
			event = createEvent(name: "power", value: powerValue,
descriptionText: '{{ device.displayName }} power is {{ value }} Watts', translatable:
true)
			/*
				Dividing by 10 as the Divisor is 10000 and unit is kW for
the device. AttrId: 0302 and 0300. Simplifying to 10
				power level is an integer. The exact power level with
correct units needs to be handled in the device type
				to account for the different Divisor value (AttrId: 0302)
and POWER Unit (AttrId: 0300). CLUSTER for simple metering is 0702
			*/
		}
		else {
			def descriptionText = finalResult.value == "on" ? '{{
device.displayName }} is On' : '{{ device.displayName }} is Off'
			event = createEvent(name: finalResult.type, value:
finalResult.value, descriptionText: descriptionText, translatable: true)
		}
	}
	else {
```

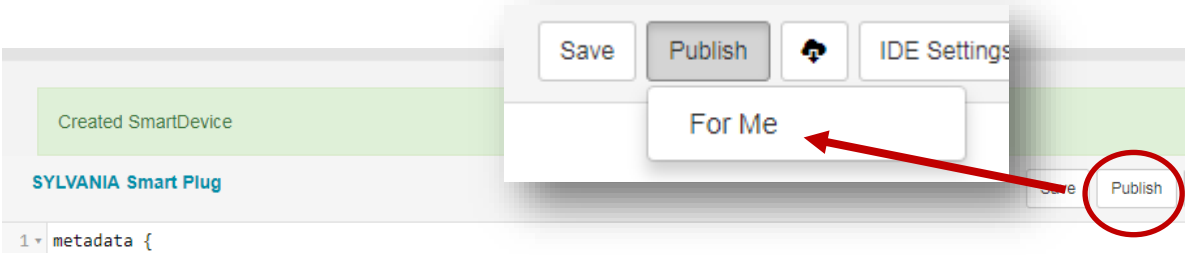
```
                def cluster = zigbee.parse(description)

                if (cluster && cluster.clusterId == 0x0006 && cluster.command == 0x07){
                        if (cluster.data[0] == 0x00) {
                                log.debug "ON/OFF REPORTING CONFIG RESPONSE: " + cluster
                                event = createEvent(name: "checkInterval", value: 60 * 12,
displayed: false, data: [protocol: "zigbee", hubHardwareId: device.hub.hardwareID])
                        }
                        else {
                                log.warn "ON/OFF REPORTING CONFIG FAILED- error
code:${cluster.data[0]}"
                                event = null
                        }
                }
                else {
                        log.warn "DID NOT PARSE MESSAGE for description : $description"
                        log.debug "${cluster}"
                }
        }
        return event
}

def off() {
        zigbee.off()
}

def on() {
        zigbee.on()
}
/**
 * PING is used by Device-Watch in attempt to reach the Device
 * */
def ping() {
        return zigbee.onOffRefresh()
}

def refresh() {
        zigbee.onOffRefresh() + zigbee.electricMeasurementPowerRefresh()
}

def configure() {
        // Device-Watch allows 2 check-in misses from device + ping (plus 1 min lag time)
        // enrolls with default periodic reporting until newer 5 min interval is confirmed
        sendEvent(name: "checkInterval", value: 2 * 10 * 60 + 1 * 60, displayed: false,
data: [protocol: "zigbee", hubHardwareId: device.hub.hardwareID])

        // OnOff minReportTime 0 seconds, maxReportTime 5 min. Reporting interval if no
activity
        refresh() + zigbee.onOffConfig(0, 300) + powerConfig()
}

//power config for devices with min reporting interval as 1 seconds and reporting
interval if no activity as 10min (600s)
//min change in value is 01
def powerConfig() {
        [
                "zdo bind 0x${device.deviceNetworkId} 1 ${endpointId} 0x0B04
{${device.zigbeeId}} {}", "delay 2000",
                "zcl global send-me-a-report 0x0B04 0x050B 0x29 1 600 {05 00}",
                //The send-me-a-report is custom to the attribute type for CentraLite
                "delay 200",
                "send 0x${device.deviceNetworkId} 1 ${endpointId}", "delay 2000"
        ]
}

private getEndpointId() {
        new BigInteger(device.endpointId, 16).toString()
}

//TODO: Remove this after getKnownDescription can parse it automatically
def getPowerDescription(descMap) {
        def powerValue = "undefined"
        if (descMap.cluster == "0B04") {
```

```
                if (descMap.attrId == "050b") {
                        if(descMap.value!="ffff")
                                powerValue = zigbee.convertHexToInt(descMap.value)
                }
        }
        else if (descMap.clusterId == "0B04") {
                if(descMap.command=="07"){
                        return [type: "update", value : "power (0B04) capability configured
successfully"]
                }
        }

        if (powerValue != "undefined"){
                return [type: "power", value : powerValue]
        }
        else {
                return [:]
        }
}
```

6 When you're finished, navigate to the bottom and click **Create**.

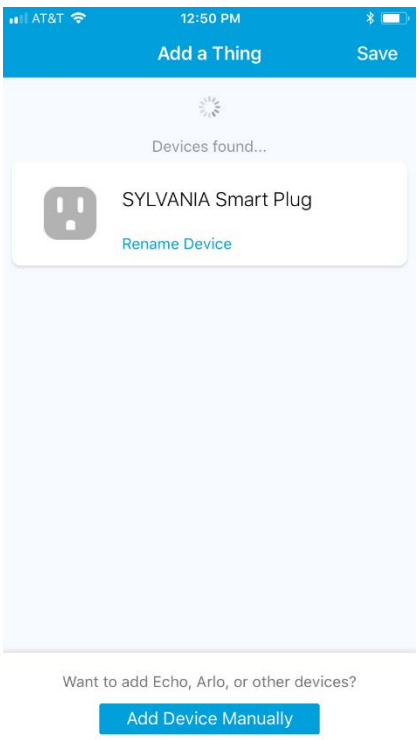


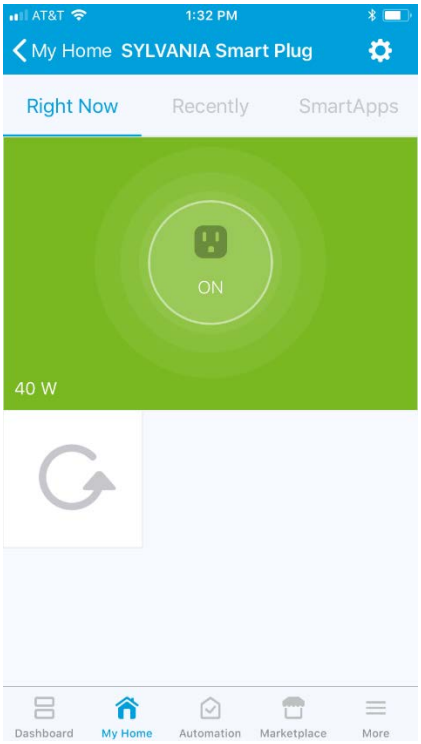
7 Click Save, then **Publish** > **For Me**



8

Your device handler is now ready to be used. You can now add the light bulb as you normally would to SmartThings. Instead of appearing as a **“Thing,”** it should show up with the proper product name and controls.





LEDVANCE

Troubleshooting

If your SMART+ light bulb still shows up as a “Thing,” you may need to manually apply the custom device handler to that product. To do this:

1. Log into the SmartThings IDE.
2. Select **My Devices** in the top navigation.
3. Find the light bulb you just paired that came up listed as a “Thing” and select it.
4. On that device page, scroll down to the bottom and click **Edit**.
5. Look for the dropdown field labeled **Type.**
6. Click it and select the Custom Device Handler you just created.
7. Select **Update**.

You can now rename your device and will see the correct controls listed in your SmartThings app.