SmartThings Custom Device Type for Somfy Controller (ZRTSI)

```
/**
 *  Somfy Blinds Controller
 *
 *  Copyright 2014 Chris Wood
 *
 *  Licensed under the Apache License, Version 444 (the "License"); you may not use this file except
 *  in compliance with the License. You may obtain a copy of the License at:
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 *  Unless required by applicable law or agreed to in writing, software distributed under the License is
distributed
 *  on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License
 *  for the specific language governing permissions and limitations under the License.
 *
 */
metadata {
     definition (name: "Somfy ZRTSI Controller", namespace: "chriswood1001", author: "Chris Wood") {
          fingerprint deviceId: "0x0200", inClusters: "0x86, 0x72"
     }

     simulator {

     }

     tiles {
          standardTile("state", "device.state", width: 2, height: 2) {
               state 'connected', icon: "st.unknown.zwave.static-controller", backgroundColor:"#ffffff"
          }

          main "state"
          details(["state"])
     }
}

def parse(String description) {
     def result = null
```

```
    if (description.startsWith("Err")) {
        result = createEvent(descriptionText:description, displayed:true)
    } else {
        def cmd = zwave.parse(description)
        if (cmd) {
            result = createEvent(zwaveEvent(cmd))
        }
    }
    return result
}

def zwaveEvent(physicalgraph.zwave.commands.securityv1.SecurityMessageEncapsulation cmd) {
    def event = [displayed: true]
    event.linkText = device.label ?: device.name
    event.descriptionText = "$event.linkText: ${cmd.encapsulatedCommand()} [secure]"
    event
}

def zwaveEvent(physicalgraph.zwave.Command cmd) {
    def event = [displayed: true]
    event.linkText = device.label ?: device.name
    event.descriptionText = "$event.linkText: $cmd"
    event
}
```

SmartThings Custom Device Type for Somfy Blinds

```
/**
 *  Somfy Blinds Controller
 *
 *  Copyright 2014 Chris Wood
 *
 *  Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except
 *  in compliance with the License. You may obtain a copy of the License at:
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
```

```
 *  Unless required by applicable law or agreed to in writing, software distributed under the License is
distributed
 *  on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License
 *  for the specific language governing permissions and limitations under the License.
 *
 */
metadata {
     definition (name: "Somfy Blind", namespace: "chriswood1001", author: "Chris Wood") {
          capability "Switch"
          capability "Switch Level"
          capability "Refresh"

          fingerprint deviceId: "0x1105", inClusters: "0x2C, 0x72, 0x26, 0x20, 0x25, 0x2B, 0x86"

         // 0x25 - Switch Binary         - Control of on/off switches
         // 0x26 - Switch Multilevel     - control of dimmer switches
           // ON   is blinds fully up   (ie. rollers fully up, slates closed pointing up)
           // OFF  is blinds fully down (ie. rollers fully down, slates closed pointing down)
           // STOP is blinds set to the Somfy 'my' position
         // 0x72 - Manufacturer Specific - report manufacturer and model (via. numeric code)
         // 0x20 - Basic                 - a generalized get/set/report command class that all devices support.
         // 0x86 - Version               - All devices report their Z-Wave framework and firmware version.
         // 0x2B - Scene Activation      - Unknown
         // 0x2C - Scene Actuator Conf   - Unknown
     }

     simulator {
          // status messages
     }

     tiles {
          standardTile("switch", "device.switch", width: 2, height: 2, icon: "st.Home.home9", canChangeIcon: true) {
              state "open", label:'open', action:"switch.off", icon:"st.Home.home9", backgroundColor:"#79b821", nextState:"closedd"
              state "closedd", label:"closed", action:"refresh.refresh", icon:"st.Home.home9", backgroundColor:"#ffffff", nextState:"open"
               state "closedu", label:"closed", action:"refresh.refresh", icon:"st.Home.home9", backgroundColor:"#ffffff", nextState:"open"
          }

          main "switch"
          details(["switch"])
     }
```

```
}

def parse(String description) {
     def result = null
    def cmd = zwave.parse(description)
    if (cmd) {
        result = zwaveEvent(cmd)
        log.debug "Parsed ${cmd} to ${result.inspect()}"
    } else {
        log.debug "Non-parsed event: ${description}"
    }
    return result
}

def zwaveEvent(physicalgraph.zwave.commands.basicv1.BasicReport cmd) {
}

def zwaveEvent(physicalgraph.zwave.commands.basicv1.BasicSet cmd) {
     [name: "switch", value: cmd.value ? "on" : "off", type: "physical"]
}

def zwaveEvent(physicalgraph.zwave.Command cmd) {
    return createEvent(descriptionText: "${device.displayName}: ${cmd}")
}



def on() {
    log.debug "Closing blinds in the 'up' position"
    [name: "switch", value: "closedu", type: "physical"]
    return zwave.switchMultilevelV1.switchMultilevelSet(value: 0xFF).format()
}

def off() {
     log.debug "Closing blinds in the 'down' position"
     [name: "switch", value: "closedd", type: "physical"]
     return zwave.switchMultilevelV1.switchMultilevelSet(value: 0x00).format()
}
```

```
def refresh() {
     log.debug "Opening blinds to 'my' position"
     [name: "switch", value: "open", type: "physical"]
     return zwave.switchMultilevelV1.switchMultilevelStopLevelChange().format()
}
```