

vDC API - vdSD properties

Contents

| | |
|--|-----------|
| Basics | 2 |
| Virtual digitalSTROM device (vdSD) properties | 2 |
| <i>Properties related to the entire device</i> | <i>2</i> |
| <i>Button Inputs</i> | <i>3</i> |
| <i>Binary Inputs</i> | <i>5</i> |
| <i>Outputs</i> | <i>7</i> |
| <i>Sensors</i> | <i>9</i> |
| <i>Scenes</i> | <i>11</i> |
| digitalSTROM 1.0 mapping compatibility | 13 |
| <i>2-way buttons</i> | <i>13</i> |
| <i>Multiple vdSDs in a single hardware device</i> | <i>13</i> |

Basics

- This document is based on the "vdSM vDC API" specification. Please refer to the corresponding document.
- This document specifies the properties specific to virtual devices (vdSD) managed by a virtual device controller (vDC).

Virtual digitalSTROM device (vdSD) properties

- The following table applies to entities which have a value of "vdSD" for the "type" property.
- All vdSDs must also support the basic set of properties as described under "Common properties" in the "vdSM vDC API" document:

- **dSUID**
- type (value is always "vdSD" for virtual devices)
- model
- hardwareVersion
- hardwareGUID
- oemGuid
- name

Properties related to the entire device

| property name | acc | Type/range | description | R105 mapping |
|--------------------------|-----|-------------------------------|---|--------------------------|
| primaryGroup | r | integer, dS group number 1..8 | basic group (color) of the device | Function ID Bits 15...12 |
| isMember[groupNo] | r/w | boolean | array of boolean flags, array index represents dS group number | Bank 1 - 0x10..0x11 |
| zoneID | r/w | integer, global dS Zone ID | this should be updated by the vdSM to reflect the zone the device is in. The vDC may use this value to optimize zone calls (i.e. bundle calls to actual hardware if single device calls are slow) | |
| localPriority | r/w | boolean | enables local priority. In local priority, device ignores scene calls unless the scene has the <i>ignoreLocalPriority</i> flag set, or the <i>callScene</i> call has the <i>force</i> parameter set to true | |

| property name | acc | Type/range | description | R105 mapping |
|------------------------|-----|------------------|--|--------------|
| progMode | r/w | boolean | enables local programming mode (for those devices that have it) | |
| numDevicesInHW | r | optional integer | Number of separate vdSDs that represent the same hardware device (which MUST have the same <i>hardwareGUID</i>) Present only if a unambiguous statement can be made about the number of devices per hardware. Devices that are <i>usually</i> grouped in one chassis, but can be taken apart should not report this property. | |
| deviceIndexInHW | r | optional integer | only if <i>numDevicesInHW</i> exists, this enumerates the devices within the same hardware | |

Button Inputs

| property name | acc | Type/range | description | R105 mapping |
|----------------------------------|-----|---|--|---------------------|
| buttonInputDescriptions[] | r | object | array of object, representing capabilities of button inputs | Bank 1 - 0x40..0x57 |
| name | r | string | human readable name/ number for the input (e.g. matching labels for hardware connectors) | |
| supportsLocalKeyMode | r | boolean | can be local button | hardware |
| buttonID | r | integer 0..n (optional) | ID of physical button. No ID means no fixed assignment to a button. All elements of a multi-function hardware button must have the same buttonID. | hardware |
| buttonType | r | integer enum (inputs with buttons supported only) | Type of physical button 0: undefined 1: single pushbutton 2: 2-way pushbutton 3: 4-way navigation button 4: 4-way navigation with center button 5: 8-way navigation with center button 6: on-off switch | hardware |

| property name | acc | Type/range | description | R105 mapping |
|------------------------------|-----|---|--|--------------------------------------|
| buttonElementID | r | integer (inputs with buttons supported only) | Element of multi-contact button: 0: center 1: down 2: up 3: left 4: right 5: upper left 6: lower left 7: upper right 8: lower right Note: For undefined <i>buttonType</i> , <i>buttonElement</i> just enumerates the elements (0..numElements-1) | hardware |
| buttonInputSettings[] | r/w | object | array of objects, representing configuration settings of buttons and binary inputs | Bank 3 - 0x01 Bank 1 - 0x40..0x57 |
| group | r/w | integer | dS group number 1..8 | Bank 3 - 0x01 Bank 1 - 0x40..0x57 |
| function | r/w | integer 0..15 | see LTNUM descriptions (0: device, 5: room, ...) | LTNUM |
| mode | r/w | integer | 255: inactive 0: standard 2: presence 5..8 : button1..4 down 9..12 : button1..4 up | LTMODE |
| setsLocalPriority | r/w | boolean | button should set local priority | |
| callsPresent | r/w | boolean | button should call present (if system state is absent) | |
| buttonInputStates[] | r | object | representation of the current state of the button | Bank 64 - 0x01 |
| value | r | boolean or NULL | false=inactive, true=active, NULL=unknown state | |

| property name | acc | Type/range | description | R105 mapping |
|---------------|-----|----------------|---|--------------|
| clickType | r | integer enum | Most recent click state of the button: 0: tip_1x 1: tip_2x 2: tip_3x 3: tip_4x 4: hold_start 5: hold_repeat 6: hold_end 7: click_1x 8: click_2x 9: click_3x 10: short_long 11: local_off 12: local_on 13: short_short_long 14: local_stop 255: idle (no recent click) | |
| age | r | double or NULL | age of the state shown in the <i>value</i> and <i>clickType</i> fields in seconds. If no recent state is known, returns NULL. | |
| error | r | integer enum | 0: ok 1: open circuit 2: short circuit 4: bus connection problem 5: low battery in device 6: other device error | |

Binary Inputs

| property name | acc | Type/range | description | R105 mapping |
|----------------------------------|-----|--|--|---------------------|
| binaryInputDescriptions[] | r | object | array of object, representing capabilities of binary inputs | Bank 1 - 0x40..0x57 |
| name | r | string | human readable name/ number for the input (e.g. matching labels for hardware connectors) | |
| inputType | r | integer (inputs with binary functions supported only) | 0: poll only 1: detects changes | |

| property name | acc | Type/range | description | R105 mapping |
|------------------------------|-----|--------------|--|--------------------------------------|
| inputUsage | r | integer enum | Describes the usage field for the input (beyond device color) 0: undefined (generic usage or unknown) 1: room climate 2: outdoor climate 3: climate setting (from user) | |
| hardwareSensorFunction | r | integer enum | hardwired function of this input if it is not freely configurable. See sensorFunction in binaryInputSettings[] below for possible values. 0 means generic input with no hardware-defined functionality. | |
| updateInterval | r | double | how fast the physical value is tracked, in seconds | |
| binaryInputSettings[] | r/w | object | array of objects, representing configuration settings of buttons and binary inputs | Bank 3 - 0x01 Bank 1 - 0x40..0x57 |
| group | r/w | integer | dS group number 1..8 | Bank 3 - 0x01 Bank 1 - 0x40..0x57 |
| binaryMode | r/w | integer enum | 0 disabled (no push) 0x10 standard 0x11 inverted 0x12 rising edge on 0x13 falling edge on 0x14 rising edge off 0x15 falling edge off 0x16 rising edge 0x17 falling edge | |

| property name | acc | Type/range | description | R105 mapping |
|----------------------------|-----|-----------------|---|----------------|
| sensorFunction | r/w | integer enum | 0x00 App Mode (no system function) 0x01 Presence (Präsenz) 0x02 Light (Helligkeit) – aktuell noch nicht in Verwendung 0x03 Presence in darkness (Präsenz bei Dunkelheit) – aktuell noch nicht in Verwendung 0x04 Twilight (Dämmerung) 0x05 Motion detector (Bewegung) 0x06 Motion in darkness (Bewegung bei Dunkelheit) – aktuell noch nicht in Verwendung 0x07 Smoke detector (Rauchmelder) 0x08 Wind monitor (Windwächter) 0x09 Rain monitor (Regenwächter) 0x0a Solar radiation (Sonneneinstrahlung) 0x0b Thermostat (Thermostat) | |
| binaryInputStates[] | r | object | representation of the current state of the inputs | Bank 64 - 0x01 |
| value | r | boolean or NULL | false=inactive, true=active, NULL=undefined | |
| age | r | double or NULL | age of the state shown in the <i>value</i> field in seconds. If no recent state is known, returns NULL | |
| error | r | integer enum | 0: ok 1: open circuit 2: short circuit 4: bus connection problem 5: low battery in device 6: other device error | |

Outputs

| property name | acc | Type/range | description | R105 mapping |
|-----------------------------|-----|------------|--|--------------|
| outputDescriptions[] | r | object | array of objects, representing hardware capabilities of output | |

| property name | acc | Type/range | description | R105 mapping |
|-------------------------|-----|--------------|---|---|
| name | r | string | human readable name/ number for the output (e.g. matching labels for hardware connectors) | |
| function | r | integer enum | 0: on/off only 1: dimmer 2: positional | hardware |
| outputUsage | r | integer enum | Describes the usage field for the output (beyond device color) 0: undefined (generic usage or unknown) 1: room 2: outdoors 3: user (display/indicator) | |
| variableRamp | r | boolean | supports variable ramps | Function-ID Bit 5 |
| maxPower | r | integer | max output power in Watts. If absent, power capability is undefined | hardware |
| minDim | r | integer | minmum brightness that hardware supports (for dimming outputs) | hardware |
| outputSettings[] | r/w | object | array of objects, representing operation mode of output | |
| group | r/w | integer | dS group number 1..8 | Bank 3 - 0x01 Bank 1 - 0x40..0x57 |
| mode | r/w | integer enum | 0: disabled, inactive 1: binary 2: gradual | |
| pushChanges | r/w | boolean | if set, locally generated changes in the output value will be pushed | |
| dimTimeUp | r/w | integer | dim up time in ms | Bank 3 - 0x06 |
| dimTimeDown | r/w | integer | dim down time in ms | Bank 3 - 0x07 |
| dimTimeUpAlt1 | r/w | integer | alternate 1 dim up time in ms | Bank 3 - 0x08 |
| dimTimeDownAlt1 | r/w | integer | alternate 1 dim down time in ms | Bank 3 - 0x09 |
| dimTimeUpAlt2 | r/w | integer | alternate 2 dim up time in ms | Bank 3 - 0x10 |
| dimTimeDownAlt2 | r/w | integer | alternate 2 dim down time in ms | Bank 3 - 0x11 |

| property name | acc | Type/range | description | R105 mapping |
|-----------------------|-----|--------------|---|----------------|
| outputStates[] | r/w | object | array of output states | Bank 64 - 0x00 |
| value | r/w | integer | current output value (brightness, blind position, on/off) | |
| age | r | double | age of the state shown in the <i>value</i> field in seconds. This indicates when the value was last applied to the actual device hardware, or when an actual output status was last received from the device. <i>age</i> is NULL when a new value was set, but not yet applied to the device | |
| error | r | integer enum | 0: ok 1: open circuit / lamp broken 2: short circuit 3: overload 4: bus connection problem 5: low battery in device 6: other device error | |

Sensors

| property name | acc | Type/range | description | R105 mapping |
|-----------------------------|-----|--------------|--|---------------------|
| sensorDescriptions[] | r | object | description of sensor capabilities | Bank 1 - 0x20..0x3f |
| name | r | string | human readable name/ number for the sensor | |
| sensorType | r | integer enum | Describes the type of physical unit the sensor measures 0 : none 1 : temperature in °C 2 : relative humidity in % 3 : illumination in lux 4 : supply voltage level in V 5 : CO concentration in ppm 6 : Radon activity in Bq/m3 7 : gas type sensor 8 : particles <10µm in µg/m3 9 : particles <2.5µm in µg/m3 10 : particles <1µm in µg/m3 11 : room operating panel set point, 0..1 12 : fan speed, 0..1 (0=off, <0=auto) 13 : wind speed in m/s | |

| property name | acc | Type/range | description | R105 mapping |
|--|-----|--|---|----------------------------|
| sensorUsage | r | integer enum | Describes the usage field for the sensor 0: undefined (generic usage or unknown) 1: room 2: outdoor 3: user (setting, dial) | |
| min | r | double | min value | |
| max | r | double | max value | |
| resolution | r | double | resolution (size of LSB of actual HW sensor) | |
| updateInterval | r | double | how fast the physical value is tracked, in seconds | |
| sensorSettings[] | r/w | object | sensor configuration | |
| group | r/w | integer | dS group number 1..8 | |
| minPushInterval | r/w | double | minimum interval between pushes of changed state in seconds | |
| Note: trigger related fields are draft only - details tbd. | | | | |
| triggerLevel | r/w | double | trigger level for sensor action | ET[] Byte 1/2 |
| triggerPushDelta | r/w | double | minimum change in sensor value (in sensor units) required to trigger a state push or sensor action | ET[] Byte2/3 |
| triggerCondition | r/w | integer enum | 0: equal 1: sensor below trigger 2: sensor above trigger | |
| triggerScene | r/w | integer (optional if trigger should call scene) | scene number to call on action | ET[] Byte 5, Byte0.Bit0/1 |
| triggerButtonID | r/w | integer (optional if trigger should simulate button press) | button ID to use for simulated button action | ET[] Byte 5, Byte0.Bit0/1 |
| triggerButtonClick | r/w | integer (only when triggerButtonID is set) | clickType to use for simulated button action | ET[] Byte 5, Byte0.Bit0/1 |
| sensorStates[] | r | object | sensor states | Bank 1 - 0x40 ff Bank 6 |

| property name | acc | Type/range | description | R105 mapping |
|---------------|-----|----------------|--|--------------|
| value | r | double or NULL | current sensor value in the unit specified in SensorCapabilities.unit If no recent state is known, returns NULL. | |
| age | r | double or NULL | age of the state shown in the <i>value</i> field in seconds. If no recent state is known, returns NULL | |
| error | r | integer enum | 0: ok 1: open circuit 2: short circuit 4: bus connection problem 5: low battery in device 6: other device error | |

Scenes

| property name | acc | Type/range | description | R105 mapping |
|---------------------|-----|---|---|---------------------------|
| scenes[] | r/w | object | array of saved device states that can be recalled via callScene. Index is scene number | Bank 128ff |
| value | r/w | optional integer (or NULL when writing to actively delete the value from the scene) | primary output value for this scene (usually brightness). If value is not present, calling scene does not affect corresponding output value (Note that scene-level <i>dontCare</i> flag can be used to prevent applying any scene values) | SCE, SCE_LO, SCECON |
| valueN | r/w | optional integer (or NULL when writing to actively delete the value from the scene) | with N=1..x - secondary values, like blind angle etc., depending on device types. If value is not present, calling scene does not affect corresponding output value (Note that scene-level <i>dontCare</i> flag can be used to prevent applying any scene values) | Bank130, ScnAngle |
| dontCare | r/w | boolean | calling this scene does not apply the stored output values | SCECON |
| ignoreLocalPriority | r/w | boolean | calling this scene overrides local priority | SCECON |

| property name | acc | Type/range | description | R105 mapping |
|--------------------------|-----|--------------|--|--------------|
| flashing | r/w | boolean | calling this scene blinks the output | SCECON |
| dimTimeSelector | r/w | integer 0..2 | selects the dimming time: <ul style="list-style-type: none"> • 0 = use <i>dimTimeUp/</i> <i>dimTimeDown</i> • 1 = use <i>dimTimeUpAlt1/</i> <i>dimTimeDownAlt1</i> • 2 = use <i>dimTimeUpAlt2/</i> <i>dimTimeDownAlt2</i> | SCECON |
| <i>other_scene_value</i> | | | scenes might contain additional device specific scene values not currently used by the dS system | |

digitalSTROM 1.0 mapping compatibility

An important design goal for the vDC API and the vdSD property set was to avoid carrying over dS 1.0 specific limitations.

On the other hand, the vDC API was designed to support capabilities current dSS 1.x architecture can't support yet, but are likely to be implemented in future dS versions.

Still, the vDC + vdSM needs to be compatible with existing dSS 1.x installations.

To achieve this, vDC devices (vdSDs) that provide functionality similar or equal to existing hardware digitalSTROM devices (dSDs), must have **sensible default settings that make them mappable into existing dSS 1.x installations**.

This chapter lists the conventions that must be followed for certain device types to make them mappable into dSS 1.x environments.

2-way buttons

2-way buttons (rockers) like present in many enOcean devices must conform to the following default behaviour:

1. the vdSD must have two button inputs (represented by 2 array elements in the *buttonInputDescriptions/Settings/States* property arrays)
2. the buttonInput with index = 0 must represent the "down" button
3. the buttonInput with index = 1 must represent the "up" button
4. buttonInputSettings[0].mode must be 6 (down button paired with second input)
5. buttonInputSettings[1].mode must be 9 (up button paired with first input)
6. in the *dSUID* space, the *dSUID* following the *dSUID* of the device (device's *dSUID* + 1) must be guaranteed unused. This means that the *idBlockSize* property must be 2 to document that the next *dSUID* is guaranteed unused. This *allows* the vdSM to virtually split the device into two separate *dSUIDs* to mimic for example a SW-TKM210 towards the dSS 1.x environment. The vdSM *may* also choose to represent the device as a single *dSUID* with a inseparable 2-way button instead, like a GR-TKM210.

Multiple vdSDs in a single hardware device

Some hardware devices contain more than one instance of a certain functional unit. Usually, these are represented as a separate vdSD each, to allow maximum flexibility in the way the functional units can be used.

For example, a dual 2-way button enOcean device will be represented as 2 entirely separate vdSDs, because despite the physical proximity, each button might control a different zone, group or function. By default, such a device will be represented as 2 separate SW-TKM210 (dual input) devices. However, the vdSM might want to represent it as a single SW-TKM200 (quad input) device. To allow the vdSM to find out which and how many vdSDs are in the same hardware device, the vdSD *should* expose this information in the *numDevicesInHW* and *deviceIndexInHW* properties as follows:

1. *numDevicesInHW* contains the number of vdSDs in the same hardware device
2. *hardwareGUID* identifies the hardware device of which the vdSD is part of

3. *deviceIndexInHW* contains an index, $0..numDevicesInHW-1$ that enumerates the vdSDs in the same hardware device
4. This association of vdSDs to a containing hardware device must only be made when the number of grouped vdSDs and enumeration is unambiguous and permanent. So just 3 modules that usually ship mounted on a common frame, but can be easily separated and used independently should not have *numDevicesInHW* and *deviceIndexInHW* properties.