vDC API properties

Contents

API Version History

This document covers vDC API Version 2

2 cleaned up, added multiple vdcs per vdc host, added MOC support, new and more powerful property access methods
1 initial version of vDC API (shown as “1.0” in JSON API)

Basics

- This document is based on the "vDC API" specification. Please refer to the corresponding document for general description of the API and the available messages/ calls.
- This document specifies the named properties available for different types of *addressable entities* (vDC, vdSD, vDC host) as well as the properties common for all types of *addressable entities.*
- All strings are UTF-8 encoded
- Properties marked optional may or may not be available in a particular implementation. If not available, getProperty will just not return them in the result tree, but will NOT return an error response (see description of *getProperty* in the “vDC API” specification for details). However, a setProperty call containing a value for a non-implemented property will return an error.

Common properties for all addressable entities

- The common properties must be supported by entities which can be addressed via a dSUID using this API (*addressable entities*). At the time of writing, this includes virtual devices, logical vDCs and the vDC host, but may be extended to other entities.
- The "type" property reveals the kind of entity.
- Properties common to a specific entity type are maintained in separate documents. In particular, see "vdSD properties" document for common properties of virtual devices.

property name	acc	Type/range	description
dSUID	r	string of 34 hex characters (2*17)	the dSUID of the entity. Normally not used in regular vDC API calls, as these require the dSUID in the getProperty() call. Useful for debugging.

property name	acc	Type/range	description
idBlockSize	r	optional integer	The number of consecutive dSUIDs (starting with this device's dSUID) that are reserved. This means that the next *idBlockSize*-1 dSUIDs are guaranteed not being used by any vdSD in any vDC. This mechanism allows that a vdSM can implement representing multi-input devices as multiple dSUIDs for backwards compatibility with dS 1.0
type	r	string	Type of entity: "vdSD" : virtual dS device "vDC" : a logical vDC "vDChost" : the vDC host "vdSM" : a vdSM
model	r	string	(short, language independent) human-readable model string of the entity. Should be descriptive enough to allow a human to associate it with a *kind* of hardware or software. Is mapped to “hardwareInfo” in vdsm and upstream
hardwareVersion	r	optional string	string describing the hardware device's version represented by this entity, if available
hardwareGuid	r	optional string	hardware's native globally unique identifier (GUID), if any, in URN-like format: *formatname*:*actualID* The following formats are in use: • *enoceanaddress:XXXXXXXX* = 8 hex digits EnOcean device address • *gs1:(01)ggggg(21)sssss* = GS1 formatted GTIN plus Serial number • *uuid:UUUUUUU* = UUID • *macaddress:MMMMM* = MAC Address
modelGuid	r	optional string	model’s native globally unique identification, if any, in URN-like format: *formatname*:*actualID* The following formats are in use: • *enoceaneep:ooffft* = 6 hex digits EnOcean Equipment Profile (EEP) number • *gs1:(01)ggggg* = GS1 formatted GTIN

property name	acc	Type/range	description
vendorGuid		optional string	globally unique identification of the vendor, in URN-like format: The following formats are in use: • *enoceanvendor:vvv[:name]* = 3 hex digits EnOcean vendor ID, optionally followed by a colon and the clear text vendor name if known • *vendorname:name* = clear text name of the vendor • *gs1:(412)lllll* = GS1 formatted Global Location Number of the vendor
oemGuid	r	optional string	Globaly unique identifier (GUID) of the product the hardware is embedded in, if any - see hardwareGuid for format variants.
configURL	r	optional string	full URL how to reach the web configuration of this device (if any)
deviceIcon16	r	optional binary png image	16x16 pixel png image to represent this device in the digitalSTROM configurator UI
deviceIconName	r	optional string	filename-safe name for the icon (a-z, 0-9, _, -, no spaces or funny characters!). This allows for more efficient caching in Web UIs - many devices might have the same icon, so web UIs don't need to load the actual data (*deviceIcon16*) for every device again, as long as devices show the same *deviceIconName.*
name	r/w	string	user-specified name of the entity. Is also stored upstreams in the vdSM and further up, but is useful for the vDC to know for configuration and debugging. The vDC usually generates descriptive default names for newly connected devices. When the vdSM registers a device, it should read this property and propagate the name towards the dSS. When the user changes the name via the dSS configurator, this property should be updated with the new name.

Virtual device connector (vDC) properties

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

Properties on the vdc level

property name	acc	Type/range	description
capabilities	r	list of property elements	Descriptions (invariable properties) of the vdc's capabilities. • each capability is represented as a property element (key value) See below for defined capabilities
zoneID	r/w	integer, global dS Zone ID	this should be updated by the vdSM to reflect the default zone the vdc has.

VDC Capabilities

Capabilities of the vDC. The properties described here are contained in the vdc-level *capabilities* property.

property name	acc	Type/range	description
metering	r	optional boolean	if true, the vdc provides metering data

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" above.

Properties on the device level

General device properties

property name	acc	Type/range	description
primaryGroup	r	integer, dS group number 1..11	basic group (color) of the device
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)

property name	acc	Type/range	description
progMode	r/w	optional 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 *hardwareGUID*) Present only if a unambiguous statement can be made about the number of devices per hardware. Devices that are *usually* grouped in one chassis, but can be taken apart should not report this property.
deviceIndexInHW	r	optional integer	only if *numDevicesInHW* exists, this enumerates the devices within the same hardware

Inputs

Virtual devices can have zero to several buttons, binary (digital) inputs and sensors. The following container properties provide access to the set of properties related to each input. The individual subproperties are described in separate paragraphs further down.

property name	acc	Type/range	description
buttonInputDescriptions	r	optional list of property elements	Descriptions (invariable properties) of the buttons in the device. • represented as a list of property elements, one for each button in the device. • The property elements are named sequentially “0”,”1”,... See below for details
buttonInputSettings	r/w	optional list of property elements	Settings (properties that can be changed and are stored persistently in the vDC) of the buttons in the device. See below for details
buttonInputStates	r/w	optional list of property elements	State (changing during operation of the device, but not persistently stored in the vDC) of a button. See below for details
binaryInputDescriptions	r	optional list of property elements	Descriptions of the binary inputs in the device. See below for details
binaryInputSettings	r/w	optional list of property elements	Settings (properties that can be changed and are stored persistently in the vDC) of the binary inputs in the device. See below for details
binaryInputStates	r/w	optional list of property elements	State (changing during operation of the device, but not persistently stored in the vDC) of a binary input. See below for details

property name	acc	Type/range	description
sensorDescriptions	r	optional list of property elements	Descriptions of the sensors in the device. See below for details
sensorSettings	r/w	optional list of property elements	Settings (properties that can be changed and are stored persistently in the vDC) of the sensors in the device. See below for details
sensorStates	r/w	optional list of property elements	State (changing during operation of the device, but not persistently stored in the vDC) of a sensor. See below for details

Outputs and Channels

Virtual devices in the digitalSTROM system can have a *single* output (or none at all, as for pure button devices)

The output can have one or multiple *channels.* Outputs with complex functionality like color lights or blinds usually have multiple *channels* to control different aspects of the output separately

Important Notes:

- Devices with no output at all do not have output- and channel-related properties.
- *output* is meant as “output functionality” - like a lamp, a blind, a washing machine. Such *outputs* may need multiple physical parameters to control, and thus will likely have multiple physical/electrical output lines. These multiple parameters do not count as separate *outputs*, but are called *channels* (of the single output, see below)
- If a physical device does have multiple, *independent* outputs (such as a multi-channel dimmer for example), a vDC must represent such a physical device as multiple virtual devices, with separate dSUIDs. The dSUID numbering scheme provides an enumeration field (17th byte) for that purpose.

property name	acc	Type/range	description
outputDescription	r	optional list of output description properties	Descriptions (invariable properties) of the device’s output. Devices with no output don’t have this property. See below for details
outputSettings	r/w	optional list of output settings properties	Settings (properties that can be changed and are stored persistently in the vDC) of the device’s output. Devices with no output don’t have this property. See below for details
outputState	r/w	optional list of output state properties	State (changing during operation of the device, but not persistently stored in the vDC) of the outputs. Devices with no output don’t have this property. See below for details
channelDescriptions	r	optional list of property elements	Descriptions (invariable properties) of the channels in the device. • represented as a list of property elements, one for each channel in the device. • The property elements are named acording to the channel type (or “0” for outputs not assigned to a channel) See below for details
channelSettings	r/w	optional list of property elements	Settings (properties that can be changed and are stored persistently in the vDC) of the channels in the device. See below for details
channelStates	r/w	optional list of property elements	State (changing during operation of the device, but not persistently stored in the vDC) of the channels. See below for details

Scenes

This property is available on devices having at least one output with standard behaviour defined, such as light, or shadow. Input-only devices do not support this property.

property name	acc	Type/range	description
scenes	r/w	optional list of property elements	The scene table of the device • represented as a list of property elements, one for each scene. • The property elements are named by scene number, starting with “0”. See below for details

Button Input

Button Input Description

Description (invariable properties) of a button. The properties described here are contained in the elements of the device-level *buttonInputDescriptions* property.

property name	acc	Type/range	description
name	r	string	human readable name/number for the input (e.g. matching labels for hardware connectors)
supportsLocalKeyMode	r	boolean	can be local button
buttonID	r	optional integer 0..n	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.
buttonType	r	integer enum	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
buttonElementID	r	integer	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 *buttonType*, *buttonElement* just enumerates the elements (0..numElements-1)

Button Input Settings

Settings (properties that can be changed and are stored persistently in the vDC) for a button. The properties described here are contained in the elements of the device-level *buttonInputSettings* property.

property name	acc	Type/range	description
group	r/w	integer	dS group number
function	r/w	integer 0..15	see LTNUM descriptions (0: device, 5: room, ...)

property name	acc	Type/range	description
mode	r/w	integer	255: inactive 0: standard 2: presence 5..8 : button1..4 down 9..12 : button1..4 up
channel	r/w	integer enum	channel this button should control 0: (default) button controls the default channel 1..191: reserved for digitalSTROM standard channel types 192..239: device specific channel types
setsLocalPriority	r/w	boolean	button should set local priority
callsPresent	r/w	boolean	button should call present (if system state is absent)

Button Input State

State (current state, changing during operation of the device, but not persistently stored in the vDC) of a button. The properties described here are contained in the elements of the device-level *buttonInputStates* property.

property name	acc	Type/range	description
value	r	boolean or NULL	false=inactive, true=active, NULL=unknown state
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 *value* and *clickType* 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 Input

Binary Input Description

Description (invariable properties) of a binary input. The properties described here are contained in the elements of the device-level *binaryInputDescriptions* property.

property name	acc	Type/range	description
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
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)
sensorFunction	r	integer enum	hardwired function of this input if it is not freely configurable. See *sensorFunction* in *binaryInputSettings*[] below for all possible values. Specifically, hardwired functions in use are: 0 means generic input with no hardware-defined functionality. 12 Battery low status (set when battery is low)
updateInterval	r	double	how fast the physical value is tracked, in seconds

Binary Input Settings

Settings (properties that can be changed and are stored persistently in the vDC) for a button. The properties described here are contained in the elements of the device-level *binaryInputSettings* property.

property name	acc	Type/range	description
group	r/w	integer	dS group number

property name	acc	Type/range	description
sensorFunction	r/w	integer enum	0 App Mode (no system function) 1 Presence (Präsenz) 2 Light (Helligkeit) – aktuell noch nicht in Verwendung 3 Presence in darkness (Präsenz bei Dunkelheit) – aktuell noch nicht in Verwendung 4 Twilight (Dämmerung) 5 Motion detector (Bewegung) 6 Motion in darkness (Bewegung bei Dunkelheit) – aktuell noch nicht in Verwendung 7 Smoke detector (Rauchmelder) 8 Wind monitor (Windwächter) 9 Rain monitor (Regenwächter) 10 Solar radiation (Sonneneinstrahlung) 11 Thermostat (Thermostat) 12 Battery low status (set when battery is low)

Binary Input State

State (current state, changing during operation of the device, but not persistently stored in the vDC) of a button. The properties described here are contained in the elements of the device-level *binaryInputStates* property.

property name	acc	Type/range	description
value	r	boolean or NULL	false=inactive, true=active, NULL=undefined
age	r	double or NULL	age of the state shown in the *value* 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

Sensor

Sensor Description

Description (invariable properties) of a sensor input. The properties described here are contained in the elements of the device-level *sensorDescriptions* property.

property name	acc	Type/range	description
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 humitity 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 14 : Power in W 15 : Electric current in A 16 : Energy in kWh 17 : Electric Consumption in VA
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, approximately. The purpose of this is to give information about the time resolution that can be expected from that sensor.
alifeSignInterval	r	double	how fast the sensor *minimally* sends an update. If sensor does not push a value for longer than that, it can be considered out-of-order

Sensor Settings

Settings (properties that can be changed and are stored persistently in the vDC) for a sensor. The properties described here are contained in the elements of the device-level *sensorSettings* property.

property name	acc	Type/range	description
group	r/w	integer	dS group number
minPushInterval	r/w	double	minimum interval between pushes of changed state in seconds, default=2
changesOnlyInterval	r/w	double	minimum interval between pushes with same value (in case sensor hardware sends update, but with same value as before - only age will differ). default=0=all updates from hardware trigger a push

Sensor State

State (current state, changing during operation of the device, but not persistently stored in the vDC) of a sensor. The properties described here are contained in the elements of the device-level *sensorStates* property.

property name	acc	Type/range	description
value	r	double or NULL	current sensor value in the unit according to *sensorType* If no recent state is known, returns NULL.
age	r	double or NULL	age of the state shown in the *value* 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

Output

Note: devices with no output functionality return a NULL response when queried for *outputDescription*, *outputSettings* or *outputState*

Output Description

Description (invariable properties) of the device's output. The properties described here are contained in the device-level *outputDescription* property.

property name	acc	Type/range	description
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 (with channel 1, "brightness", switched on when "brigtness">"onThreshold") 1: dimmer (with channel 1, "brightness") 2: positional (blinds, for example) 3: dimmer with color temperature (with channels 1 and 4, "brightness", "ct") 4: full color dimmer (with channels 1-6, "brightness", "hue", "saturation", "ct", "cieX", "cieY"
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
maxPower	r	optional integer	max output power in Watts. If absent, power capability is undefined
minDim	r	integer	minmum brightness that hardware supports (for dimming outputs)

Output Settings

Settings (properties that can be changed and are stored persistently in the vDC) for the device's output. The properties described here are contained in the device-level *outputSettings* property.

property name	acc	Type/range	description
groups	r/w	list of boolean property elements	contains a list of property elements with a boolean value, representing this device's output's membership in one or multiple groups. • The name of the subproperty represents the dS group number ("0" to "63"). • For efficiency reasons, only "true" values are returned, so the result of requesting the entire subproperty list with a wildcard query is a list of groups the output is member of (thus usually consisting of a few elements only) • When querying a single group by ID, a NULL value is returned if the output is not a member of the queried group. • For writing, value can be true or false to add or remove a group membership.
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
onThreshold	r/w	optional integer	Light outputs: minimum brightness level that will switch on non-dimmable lamps. Defaults to 128.
dimTimeUp	r/w	optional integer	Light outputs: dim up time in ms
dimTimeDown	r/w	optional integer	Light outputs: dim down time in ms
dimTimeUpAlt1	r/w	optional integer	Light outputs: alternate 1 dim up time in ms
dimTimeDownAlt1	r/w	optional integer	Light outputs: alternate 1 dim down time in ms
dimTimeUpAlt2	r/w	optional integer	Light outputs: alternate 2 dim up time in ms
dimTimeDownAlt2	r/w	optional integer	Light outputs: alternate 2 dim down time in ms

Output State

State (current state, changing during operation of the device, but not persistently stored in the vDC) of the device's output. The properties described here are contained in the device-level outputState property.

property name	acc	Type/range	description
localPriority	r/w	boolean	enables local priority of the device's output. In local priority mode, device ignores scene calls unless the scene has the *ignoreLocalPriority* flag set, or the *callScene* call has the *force* parameter set to true
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

Channel

Channel Description

Description (invariable properties) of the device's channels. The properties described here are contained in the device-level *channelDescriptions* property.

property name	acc	Type/range	description
name	r	string	human readable name/number for the channel (e.g. matching labels for hardware connectors)
channelIndex	r	integer	0..N-1, where N=number of channels in this device. • The index is device specific and no assumption on any particular order of indexes vs. channel types must be made. • This index is **necessary** to represent the channel type mapping table as described in "Multiple Output Channels, v3, 2013-11-13" on bottom of page 5. This table is required by the vdsm to conform to upstream dS specs.
min	r	double	min value
max	r	double	max value
resolution	r	double	resolution (size of LSB of actual HW output)

Channel Settings

Settings (properties that can be changed and are stored persistently in the vDC) for the device's channels. The properties described here are contained in the device-level *channelSettings* property.

Note: currently there are no per-channel settings defined

property name	acc	Type/range	description

Channel State

State (current state, changing during operation of the device, but not persistently stored in the vDC) of the device's channels. The properties described here are contained in the device-level channelStates property.

property name	acc	Type/range	description
value	r	double	current channel value (brightness, blind position, on/off) Note: channel State must not be written to. Instead, the *setOutputChannelValue* should be used.

property name	acc	Type/range	description
age	r	double	age of the state shown in the *value* 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. *age* is NULL when a new value was set, but not yet applied to the device

Scene

A scene stores a set of values to apply to the outputs of the device when a particular scene is called.

As outputs can be looked at in two different ways, by index or by channel (see description for "Outputs" above), each scene contains the scene values for each output in two forms, once by output number (property "outputs") and once by channel type (property "channels")

property name	acc	Type/range	description
channels	r/w	list of property elements	For each channel, a scene value (consisting of *value* and *dontCare* flag, see below). • represented as a list of property elements, one for each channel in the device. • The property elements are named by channel type id (which can be 0 for devices controlling an unspecified functionality, such as a generic switch output)
effect	r/w	integer enum	Specifies the effect to be applied when this scene is invoked. The following standard effects are defined (%%% note: enum might change, specification in discussion %%%): 0 : no effect, immediate transition 1 : smooth normal transition (corresponds with former *dimTimeSelector*==0) 2 : slow transition (corresponds with former *dimTimeSelector*==1) 3 : very slow transition (corresponds with former *dimTimeSelector*==2) 4 : blink (for light devices) / alerting (in general: an effect that draws the user's attention) Notes: • stored scene values may or may not be used to parametrize the effect, depending on the type of effect. For example, the blink effect with a multi-color lamp must use the color values as set in the scene, regardless of the dontCare flags. • When the effect has finished, channels with dontCare set will revert to the value present before the effect, while channels with dontCare *not* set are expected to have now the values as stored in the scene.
dontCare	r/w	boolean	scene-global dontCare flag: if set, calling this scene does not apply *any* of the stored channel *values*, regardless of the individual scene value's dontCare flags
ignoreLocalPriority	r/w	boolean	calling this scene overrides local priority

Scene Value

A scene value contains the value to apply to the related output when the scene is called, and the dontCare flag that can be set to *prevent* the value to be applied

property name	acc	Type/range	description
value	r/w	double	The value for the related channel. The value range and resolution is the same as for the related channel's *channelState value* property
dontCare	r/w	boolean	channel-specific dontCare flag: if set, calling this scene does not apply the stored channel *value* (but possibly other channel's values which don't have dontCare set)

Control Values

Control Values are not regular properties, but like properties, control values are named values and thus *similar* to properties. Unlike properties, control values cannot be read but only written to a vdSD, using the *setControlValue* call.

control value name	Type	description
heatingLevel	double	(dS Sensortype 50): level of heating intensity -100..100: 0=no heating or cooling, 100=max heating, -100 max cooling

digitalSTROM 1.0 mapping compatibility

An important design goal for the vDC API and the 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.