# digitalSTROM Virtual-Device-Connector API properties

digitalSTROM

Version: v1.4-branch*

May 22, 2017

*Revision: 6d0a8b6a9a98d5874ebfa542a78ea62bf3c04f8d

©2017 digitalSTROM AG. All rights reserved.

The digitalSTROM logo is a trademark of the digitalSTROM. Use of this logo for commercial purposes without the prior written consent of digitalSTROM may constitute trademark infringement and unfair competition in violation of international laws.

No licenses, express or implied, are granted with respect to any of the technology described in this document. digitalSTROM retains all intellectual property rights associated with the technology described in this document. This document is intended to assist developers to develop applications that use or integrate digitalSTROM technologies.

Every effort has been made to ensure that the information in this document is accurate. digitalSTROM is not responsible for typographical errors.

digitalSTROM AG
Building Technology Park Zürich
Brandstrasse 33
CH-8952 Schlieren
Switzerland

**Even though digitalSTROM has reviewed this document, digitalSTROM MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT THIS DOCUMENT IS PROVIDED "AS IS", AND YOU, THE READER ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.**

**IN NO EVENT WILL DIGITALSTROM BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS DOCUMENT, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.**

**THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. NO DIGITALSTROM AGENT OR EMPLOYEE IS AUTHORIZED TO MAKE ANY MODIFICATION, EXTENSION, OR ADDITION TO THIS WARRANTY.**

# Contents


**1 Basics** **5**

**2 Common properties for all addressable entities** **6**

**3 Virtual device connector (vDC) properties** **8**
3.1 Properties on the vDC level . . . 8
3.2 vDC Capabilities . . . 8

**4 Virtual digitalSTROM device (vdSD) properties** **9**
4.1 Properties on the vdSD level . . . 9
4.1.1 General device properties . . . 9
4.1.2 Inputs . . . 10
4.1.3 Outputs and Channels . . . 12
4.1.4 Scenes . . . 14
4.2 Button Input . . . 15
4.2.1 Button Input Description . . . 15
4.2.2 Button Input Settings . . . 15
4.2.3 Button Input State . . . 17
4.3 Binary Input . . . 18
4.3.1 Binary Input Description . . . 18
4.3.2 Binary Input Settings . . . 19
4.3.3 Binary Input State . . . 20
4.4 Sensor Input . . . 21
4.4.1 Sensor Input Description . . . 21
4.4.2 Sensor Input Settings . . . 23
4.4.3 Sensor Input State . . . 23
4.5 Action Descriptions . . . 24
4.5.1 Parameter Objects . . . 24
4.5.2 Device Action Descriptions . . . 24
4.5.3 Standard and Custom Actions . . . 24
4.6 States and Properties . . . 26
4.6.1 Device State Descriptions . . . 26
4.6.2 Device State Values . . . 26
4.6.3 Device Property Descriptions . . . 26
4.6.4 Device Property Values . . . 27
4.7 Device Events . . . 28
4.7.1 Device Event Descriptions . . . 28
4.8 Output . . . 29
4.8.1 Output Description . . . 29
4.8.2 Output Settings . . . 30
4.8.3 Output State . . . 31
4.9 Output Channel . . . 32
4.9.1 Output Channel Description . . . 32
4.9.2 Output Channel Settings . . . 32
4.9.3 Output Channel State . . . 32
4.10 Scene . . . 34
4.10.1 Scene Value . . . 35
4.11 Control Values . . . 36

**5 digitalSTROM mapping compatibility** **37**
5.1 2-way buttons . . . 37
5.2 Multiple vdSDs in a single hardware device . . . 37

**6 Change Log** **38**

## 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.
- All names and fields are language independent, e.g. they must not change if the user selected a different system language on the VDC host system.

| 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. |
| **type** | r | string | Type of entity:<br>vdSD" : virtual dS device<br>vDC" : a logical vDC<br>vDChost" : the vDC host<br>vdSM" : a vdSM |
| **model** | r | string | 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 |
| **modelVersion** | r | string | Human-readable model version string of the device, if available |
| **modelUID** | r | string | digitalSTROM system unique ID for the functional model of the entity.<br>• *modelUID* must be equal between all functionally identical entities (especially, devices) dS system.<br>• If different connected hardware devices provide EXACTLY the same dS functionality, these devices MAY have the same *modelUID* but will have different *hardwareModelGuid*.<br>• Vice versa, for example two identical hardware input devices will have the same *hardwareModelGuid*, but different *modelUID* if one input is mapped as a button, and the other as a binaryInput. |
| **modelVersion** | r | optional string | string describing the model's version as seen by the end user (usually the firmware version of the vdc host) |
| **hardwareVersion** | r | optional string | Human-readable 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*<br><br>The following formats are in use:<br>• *gs1:(01)ggggg(21)sssss* = GS.1 formatted GTIN plus serial number<br>• *macaddress:MMMMM* = MAC Address<br>• *enoceanaddress:XXXXXXXX* = 8 hex digits EnOcean device address<br>• *uuid:UUUUUUU* = UUID |

*continued on next page*

*continued from previous page*

| Property Name | acc | Type / Range | Description |
|---|---|---|---|
| **hardwareModelGuid** | r | optional string | hardware model's native globally unique identification, if any, in URN-like format: *formatname:actualID*<br><br>The following formats are in use:<br>• *gs1:(01)ggggg* = GS.1 formatted GTIN<br>• *enoceaneep:oofftt* = 6 hex digits EnOcean Equipment Profile (EEP) number |
| **vendorName** | r | optional string | Human-readable string of the device manufacturer or vendor |
| **vendorGuid** | r | optional string | globally unique identification of the vendor, in URN-like format:<br>The following formats are in use:<br>• *enoceanvendor:vvv[:name]* = 3 hex digits EnOcean vendor ID, optionally followed by a colon and the clear text vendor name if known<br>• *vendorname:name* = clear text name of the vendor<br>• *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.<br>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. |
| **deviceClass** | r | optional string | digitalSTROM defined unique name of a device class profile |
| **deviceClassVersion** | r | optional string | revision number of the device class profile |

## Virtual device connector (vDC) properties

- The following table applies to entities which have a value of "vDC" for the "type" property.
- All vDCs must also support the basic set of properties as described under 2 "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.<br>• each capability is represented as a property element (key/-value)<br>See 3.2 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 part of the vDC-level property. See property **capabilities** above.

| 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 2 "Common properties" above.

## Properties on the vdSD level

### General device properties

| Property Name | acc | Type / Range | Description |
|---|---|---|---|
| **primaryGroup** | r | integer, dS class number | basic class (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) |
| **progMode** | r/w | optional boolean | enables local programming mode (for those devices that have it) |
| **modelFeatures** | r | list of property elements | Descriptions (invariable properties) of the device model features.<br>• each available feature is represented as a property element (key/value) with a boolean true value. Not available features are not included in the list.<br>• this property represents the virtual device's row in the "visibility Matrix", and determines what dSS configurator UI features (dialogs, settings, controls) the device will have. |

### 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.<br>• represented as a list of property elements, one for each button in the device.<br>• The property elements are named sequentially "0","1",...<br>See 4.2.1 Button Input Descriptions 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.<br>See 4.2.2 Button Input Settings 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.<br>See 4.2.3 Button Input States for details |
| **binaryInputDescriptions** | r | optional list of property elements | Descriptions of the binary inputs in the device.<br>See 4.3.1 Binary Input Descriptions 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.<br>See 4.3.2 Binary Input Settings 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.<br>See 4.3.3 Binary Input States for details |
| **sensorDescriptions** | r | optional list of property elements | Descriptions of the sensors in the device.<br>See 4.4.1 Sensor Input Descriptions 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.<br>See 4.5.3 Sensor Input Settings for details |
| **sensorStates** | r/w | optional list of property elements | Value of a sensor. Changing during operation of the device, but not persistently stored in the vDC.<br>See 4.4.3 Sensor Input States for details |
| **deviceActionDescriptions** | r | optional list of property elements | Descriptions of the available action methods in the device.<br>See 4.5 Action Descriptions for details |
| **customActions** | r/w | optional list of property elements | Descriptions of the user defined actions methods in the device.<br>See 4.5 Action Descriptions for details |
| **deviceStateDescriptions** | r | optional list of property elements | Descriptions of the available state objects in the device.<br>See 4.6.1 State Descriptions for details |

| | | | |
|---|---|---|---|
| **deviceStates** | r/w | optional list of property elements | Value of the state objects.<br>See 4.6.2 State Descriptions for details |
| **devicePropertyDescriptions** | r | optional list of property elements | Descriptions of the available property objects in the device.<br>See 4.6.3 Property Descriptions for details |
| **deviceProperties** | r/w | optional list of property elements | Value of the property objects.<br>See 4.6.4 Property Descriptions for details |
| **deviceEventDescriptions** | r | optional list of property elements | Descriptions of the available events in the device.<br>See 4.7 Event Descriptions 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.

For further notes on compatibility see 5.

| 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.<br>See 4.8.1 Output Descriptions 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.<br>See 4.8.2 Output Settings 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.<br>See 4.8.3 Output States for details |
| **outputChannelDescriptions** | r | optional list of property elements | Descriptions (invariable properties) of the channels in the device. See 4.9.1 Channel Descriptions for details |
| **outputChannelSettings** | 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.<br>See 4.9.2 Channel Settings for details |
| **outputChannelStates** | r/w | optional list of property elements | State (changing during operation of the device, but not persistently stored in the vDC) of the channels.<br>See 4.9.3 Channel States for details |

The following properties have been replaced by the corresponding lines above. For consistency and compatibility reasons the name of the property is the index, and not the type.

**Notice** The following properties shall not be used for new developments or versions.

| | | | |
|---|---|---|---|
| **channelDescriptions** | r | optional list of property elements | Descriptions (invariable properties) of the channels in the device.<br>• represented as a list of property elements, one for each channel in the device.<br>• The property elements are named according to the channel type (or "0" for outputs not assigned to a channel) |

| | | | |
|---|---|---|---|
| **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. |
| **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. |

### 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<br>• represented as a list of property elements, one for each scene.<br>• The property elements are named by scene number, starting with “0”.<br>See 4.10 Scene Descriptions 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 4.1.2 *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<br>0: undefined 1: single pushbutton<br>2: 2-way pushbutton<br>3: 4-way navigation button<br>4: 4-way navigation with center button<br>5: 8-way navigation with center button<br>6: on-off switch |
| **buttonElementID** | r | integer | Element of multi-contact button<br>0: center<br>1: down<br>2: up<br>3: left<br>4: right<br>5: upper left<br>6: lower left<br>7: upper right<br>8: lower right<br>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 4.1.2 *buttonInputSettings* property.

| Property Name | acc | Type / Range | Description |
|---|---|---|---|
| **group** | r/w | integer | dS group number |
| **function** | r/w | integer 0..15 | see LTNUM descriptions<br>(0: device, 5: room, ...) |
| **mode** | r/w | integer | 255: inactive<br>0: standard<br>2: presence<br>5..8 : button1..4 down<br>9..12 : button1..4 up |
| **channel** | r/w | integer enum | channel this button should control<br>0: (default) button controls the default channel<br>1..191: reserved for digitalSTROM standard channel types<br>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 4.1.2 *buttonInputStates* property.

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

Alternatively the buttonInputState can be used to emit direct scene calls.

| Property Name | acc | Type / Range | Description |
|---|---|---|---|
| **actionId** | r | integer | scene id |
| **actionMode** | r | integer enum | 0: normal<br>1: force<br>2: undo |

## 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 4.1.2 *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<br>1: detects changes |
| **inputUsage** | r | integer enum | Describes the usage field for the input (beyond device color)<br>0: undefined (generic usage or unknown)<br>1: room climate<br>2: outdoor climate<br>3: climate setting (from user) |
| **sensorFunction** | r | integer enum | hardwired function of this input if it is not freely configurable.<br>See sensorFunction in *binaryInputSettings*[] below for all possible values.<br>Specifically, hardwired functions in use are:<br>0 means generic input with no hardware-defined functionality.<br>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 4.1.2 *binaryInputSettings* property.

| Property Name | acc | Type / Range | Description |
|---|---|---|---|
| **group** | r/w | integer | dS group number |
| **sensorFunction** | r/w | integer enum | 0 App Mode (no system function)<br>1 Presence<br>2 Light – not yet in use<br>3 Presence in darkness – not yet in use<br>4 Twilight<br>5 Motion detector<br>6 Motion in darkness– not yet in use<br>7 Smoke detector<br>8 Wind monitor<br>9 Rain monitor<br>10 Sun radiation<br>11 Thermostat<br>12 Battery low status (set when battery is low)<br>13 Window contact (set when window is open)<br>14 Door contact (set when door is open)<br>15 Window handle, status is close, open, or tilted<br>16 Garage door contact (set when garage door is open)<br>17 Sun protection<br>18 Frost detection<br>19 Heating system enabled<br>20 Heating change-over, switch between heating and cooling mode<br>21 Initialization status (set during startup or powerup of devices)<br>22 Malfunction: Connected device is broken and requires maintenance. Operation may have seized.<br>23 Service: Connected device requires maintenance. Normal operation still possible. |

### 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 4.1.2 *binaryInputStates* property.

| Property Name | acc | Type / Range | Description |
|---|---|---|---|
| **value** | r | boolean or NULL | false=inactive<br>true=active<br>NULL=unknown state |
| **extendedValue** | r | integer or NULL | The property 'extendedValue' replaces the property 'value'.<br>If the property extendedValue is present, the property value is not needed.<br>The data from property 'value' is overwritten. |
| **age** | r | double or NULL | age of the state shown in the *value* and *clickType* fields in seconds.<br>If no recent state is known, returns NULL. |
| **error** | r | integer enum | 0: ok<br>1: open circuit<br>2: short circuit<br>4: bus connection problem<br>5: low battery in device<br>6: other device error |

## Sensor Input

### Sensor Input Description

- Description (invariable properties) of a sensor input. The properties described here are contained in the elements of the device-level 4.1.2 *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<br>0 : none<br>1 : Temperature in °C<br>2 : Relative humitity in %<br>3 : Illumination in lux<br>4 : supply voltage level in V<br>5 : CO concentration in ppm<br>6 : Radon activity in Bq/m3<br>7 : gas type sensor<br>8 : particles <10µm in µg/m3<br>9 : particles <2.5µm in µg/m3<br>10 : particles <1µm in µg/m3<br>11 : room operating panel set point, 0..100%<br>12 : fan speed, 0..1 (0=off, <0=auto)<br>13 : Wind speed in m/s (average)<br>14 : Active Power in W<br>15 : Electric current in A<br>16 : Energy Meter in kWh<br>17 : Apparent Power in VA<br>18 : Air pressure in hPa<br>19 : Wind direction in degrees<br>20 : Sound pressure level in dB<br>21 : Precipitation intensity in mm/m2 (sum of last hour)<br>22 : CO2 concentration in ppm<br>23 : Wind gust speed in m/s<br>24 : Wind gust direction in degrees<br>25 : Generated Active Power in W<br>26 : Generated Energy in kWh<br>27 : Water Quantity in l<br>28 : Water Flow Rate in l/s |

| | | | |
|---|---|---|---|
| **sensorUsage** | r | integer enum | Describes the usage field for the sensor<br>0: undefined (generic usage or unknown)<br>1: room<br>2: outdoor<br>3: user interaction (setting, dial)<br>4: device level measurement (total, sum)<br>5: device level last run<br>6: device level average |
| **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. |
| **aliveSignInterval** | 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 Input 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 4.1.2 *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<br>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).<br>default = 0 = all updates from hardware trigger a push |

### Sensor Input 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 4.1.2 *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.<br>If no recent state is known, returns NULL. |
| **contextId** | r | integer or NULL | Numerical Id of context data.<br>Optional, returns NULL if context data not available. |
| **contextMsg** | r | string or NULL | Text message of context data.<br>Optional, returns NULL if context data not available. |
| **error** | r | integer enum | 0: ok<br>1: open circuit<br>2: short circuit<br>4: bus connection problem<br>5: low battery in device<br>6: other device error |

## Action Descriptions

### Parameter Objects

- Parameter descriptions used by action method and properties

| Field Name | Attributes | Type | Description |
|---|---|---|---|
| **type** | mandatory | numeric enumeration string | data type of the parameter value |
| **min** | opt, numeric only | double | minimum value |
| **max** | opt, numeric only | double | maximum value |
| **resolution** | opt, numeric only | double | resolution (size of LSB of actual HW sensor) |
| **siunit** | opt, numeric only | string | The SI Unit as a string, incl. prefixes like kilo or milli. For examples see http://www.ebyte.it/library/educards/siunits/TablesOfSiUnitsAndPrefixes.html |
| **options** | opt, enumeration only | list of key:value pairs | the option values for the enumeration |
| **default** | opt, all types | double, string or option | the default value of this parameter |

### Device Action Descriptions

Action descriptions describe basic functionalities and operation processes of a device. They serve as a template to create custom defined actions as variation of templates with modified parameter sets.

- Description of the basic device actions methods. The properties described here are contained in the elements of the *deviceActionDescriptions* property (invariable).

| Property Name | Attributes | Type | Description |
|---|---|---|---|
| **name** | mandatory | string | name of this action property entry |
| **params** | optional | list of Parameter Objects | parameter list related to this action |
| **description** | optional | string | description of this template action |

### Standard and Custom Actions

- Standard actions are static and immutable, and as such implemented and defined by the device.
- Custom actions are dynamic. They can be created via API or can be created or changed by interaction on the device itself.
- Actions properties described here are contained in the elements of the *standardActions* property (invariable).

| Property Name | Attributes | Type | Description |
|---|---|---|---|
| **name** | mandatory | string | unique id of this standard action property entry, always has prefix *std.* |

| | | | |
|---|---|---|---|
| **action** | mandatory | string | name of the template action, on which this standard action is based upon |
| **params** | optional | list of Parameter Name : Value pairs | list of parameter values that are different to the template action |

- Actions properties described here are contained in the elements of the *customActions* property.

| Property Name | Attributes | Type | Description |
|---|---|---|---|
| **name** | mandatory | string | unique id of this custom action property entry, always has prefix *custom.* |
| **action** | mandatory | string | reference id of the template action, on which this standard action is based upon |
| **title** | mandatory | string | human readable name of this custom action, in most cases given the user |
| **params** | optional | list of Parameter Name : Value pairs | list of parameter values that are different to the template action |

### States and Properties

Device *State* represent a status within a device. *States* differ from *Properties* in the way that they have limited number of possible values, whereas *Properties* are more generic and do have limitations on their value, with respect to their type.

#### Device State Descriptions

- Description of the state. The properties described here are contained in the elements of the *deviceStateDescriptions* property (invariable).
- State changes are signaled along with the new state value.

| Property Name | Attributes | Type | Description |
|---|---|---|---|
| **name** | mandatory | string | name of this state property entry |
| **options** | mandatory | list of Option Id : Value pairs | option list related to this state, e.g. *0: Off, 1: Initializing, 2: Running, 3: Shutdown* |
| **description** | optional | string | description of this state |

#### Device State Values

- Value of the state. The properties described here are contained in the elements of the *deviceStates* property.

| Property Name | Attributes | Type | Description |
|---|---|---|---|
| **name** | mandatory | string | name of this state property entry |
| **value** | mandatory | string | option value |

#### Device Property Descriptions

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

| Property Name | Attributes | Type | Description |
|---|---|---|---|
| **name** | mandatory | string | name of this property entry |
| **type** | mandatory | numeric enumeration string | data type of the property value |
| **min** | opt, numeric only | double | minimum value |
| **max** | opt, numeric only | double | maximum value |
| **resolution** | opt, numeric only | double | resolution (size of LSB of actual HW sensor) |
| **siunit** | opt, numeric only | string | The SI Unit as a string, incl. prefixes like kilo or milli. For examples see `http://www.ebyte.it/library/educards/siunits/TablesOfSiUnitsAndPrefixes.html` |

| | | | |
|---|---|---|---|
| **options** | opt, enumeration only | list of key:value pairs | the option values for the enumeration |
| **default** | opt, all types | double, string or option | the default value of this property |

### Device Property Values

- Value of the property. The properties described here are contained in the elements of the *deviceProperties* property.

| Property Name | Attributes | Type | Description |
|---|---|---|---|
| **name** | mandatory | string | name of this state property entry |
| **value** | mandatory | string | property value |

### Device Events

Device *Events* are state-less

a status within a device. *States* differ from *Properties* in the way that they have limited number of possible values, whereas *Properties* are more generic and do have limitations on their value, with respect to their type.

#### Device Event Descriptions

- Description of the event. The properties described here are contained in the elements of the *deviceEventDescriptions* property (invariable).

| Property Name | Attributes | Type | Description |
|---|---|---|---|
| **name** | mandatory | string | name of this event property entry |
| **description** | optional | string | description of this event |

## 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 4.1.3 *outputDescription* property.

| Property Name | acc | Type / Range | Description |
|---|---|---|---|
| **defaultGroup** | r | integer | dS Application Id of the device |
| **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 "brightness">"onThreshold")<br>1: dimmer (with channel 1, "brightness")<br>2: positional (e.g. valves, blinds)<br>3: dimmer with color temperature (with channels 1 and 4, "brightness", "ct")<br>4: full color dimmer (with channels 1-6, "brightness", "hue", "saturation", "ct", "cieX", "cieY"<br>5: bipolar, with negative and positive values (e.g. combined heating/-cooling valve, in/out fan control)<br>6: internally controlled (e.g. device has temperature control algorithm integrated) |
| **outputUsage** | r | integer enum | Describes the usage field for the output (beyond device color)<br>0: undefined (generic usage or unknown)<br>1: room<br>2: outdoors<br>3: user (display/indicator) |
| **variableRamp** | r | boolean | supports variable ramps |
| **maxPower** | r | optional double | max output power in Watts. If absent, power capability is undefined |

### 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 4.1.3 *outputSettings* property.

| Property Name | acc | Type / Range | Description |
|---|---|---|---|
| **activeGroup** | r/w | integer | dS Application Id of the device |
| **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.<br>• The name of the subproperty represents the dS group number ("1" to "63").<br>• 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)<br>• When querying a single group by ID, a NULL value is returned if the output is not a member of the queried group.<br>• For writing, value can be true or false to add or remove a group membership. |
| **mode** | r/w | integer enum | 0: disabled, inactive<br>1: binary<br>2: gradual<br>127: default (generically enabled using device's default mode) |
| **pushChanges** | r/w | boolean | if set, locally generated changes in the output value will be pushed |
| **onThreshold** | r/w | optional double | Light outputs: minimum brightness level (0..100%) that will switch on non-dimmable lamps. Defaults to 50%. |
| **minBrightness** | r/w | optional double | minmum brightness (0..100%) that hardware supports (for dimming outputs). This value is used for *callSceneMin* and dimming. Devices pre-set this value according to the dimmer capabilities, but the value can be changed to adjust depending on connected light |
| **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 |
| **heatingSystemCapability** | r/w | optional integer enum | Climate control: specifies how "heatingLevel" controlValue is applied:<br>1: heating only (heatingLevel 0..100 -> output 0..100)<br>2: cooling only (heatingLevel 0..-100 -> output 0..100)<br>3: heating and cooling (heatingLevel -100..0..100 is directly applied in case of bipolar output, and absolute value (0..100) is applied to unipolar outputs) |

### 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 4.1.3 *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<br>1: open circuit / lamp broken<br>2: short circuit<br>3: overload<br>4: bus connection problem<br>5: low battery in device<br>6: other device error |

## Output Channel

### Output Channel Description

Description (invariable properties) of the device's channels. The properties described here are contained in the device-level 4.1.3 *outputChannelDescriptions* property.

| Property Name | acc | Type / Range | Description |
|---|---|---|---|
| **name** | r | string | human readable name/number for the channel (e.g. matching labels for hardware connectors) |
| **channelType** | r | integer | Numerical Type Id of the channel. For definitions of output channel types please refer to the document ds-basics.pdf, section "Output Channels". |
| **min** | r | double | min value |
| **max** | r | double | max value |

The following attributes shall no longer be used (Version 1.0.2):

| | | | |
|---|---|---|---|
| **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. |
| **resolution** | r | double | resolution (size of LSB of actual HW output) |

### 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 4.1.3 *outputChannelSettings* property.

**Notice** Currently there are no per-channel settings defined

| Property Name | acc | Type / Range | Description |
|---|---|---|---|
| **-** | - | - | - |

### Output Channel State

Current state of the device's channels. The properties described here are contained in the device-level 4.1.3 *outputChannelStates* property.

**Notice** Channel State must not be written to. Instead, the notification *setOutputChannelValue* must be used.

| Property Name | acc | Type / Range | Description |
|---|---|---|---|
| **value** | r | double | current channel value (brightness, blind position, power on/off) |
| **automatic** | r | boolean | current channel is in automatic mode |
| **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.<br>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 4.1.3 *"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 4.10.1 *Scene Value* below).<br>• represented as a list of property elements, one for each channel in the device.<br>• 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 %%%):<br><br>0 : no effect, immediate transition 1 : smooth normal transition (corresponds with former dimTimeSelector==0)<br>2 : slow transition (corresponds with former dimTimeSelector==1)<br>3 : very slow transition (corresponds with former dimTimeSelector==2)<br>4 : blink (for light devices) / alerting (in general: an effect that draws the user's attention)<br><br>Notes:<br>• 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.<br>• 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:<br>if set, calling this scene does not apply the stored channel value (but possibly other channel's values which don't have dontCare set) |
| **automatic** | r/w | boolean | channel-specific automatic flag:<br>if set, calling this scene activates the internal automatic control logic |

### 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.

| Property Name | acc | Type / Range | Description |
|---|---|---|---|
| **heatingLevel** | w | double | (dS Sensortype 50): level of heating intensity -100..100:<br>0=no heating or cooling<br>100=max heating<br>-100=max cooling |

## digitalSTROM mapping compatibility

An important design goal for the vDC API and the property set was to avoid carrying over dS 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 digitalSTROM hardware 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:

- The vdSD must have two button inputs (represented by 2 array elements in the buttonInputDescriptions/Settings/States property arrays)
- The buttonInput with index = 0 must represent the "down" button
- The buttonInput with index = 1 must represent the "up" button
- buttonInputSettings[0].mode must be 6 (down button paired with second input)
- buttonInputSettings[1].mode must be 9 (up button paired with first input)

### 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 as follows:

- The dSUID has a 17th byte reserved to enumerate devices belonging to the same hardware, starting at zero.
- The first 16 bytes of the dSUID needs to be the same for all vdSD belonging to the same hardware.
- Usually, multiple devices are enumerated 0,1,2, etc. However, in some cases, a hardware device might have different configurations with different numbers of vdSD depending on configuration - in these cases enumeration might follow other schemes than simple increment. For example, the aforementioned dual 2-way button EnOcean device uses 0 for the first and 2 for the second rocker - to possibly allow representing each rocker as two separate vdSDs (0,1 and 2,3).
- This association of vdSDs to a containing hardware device must only be made when the number of contained vdSDs and their 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 use the enumeration but have fully distinct dSUIDs (different in first 16 bytes).

# Change Log

Document History for DocumentID: digitalSTROM Virtual-Device-Connector API properties

| date | change description |
|---|---|
| **2014-12-01** | Initial version 1.0 |
| **2015-01-13** | Version 1.0.1 Removed references to non existing properties *numDevicesInHW* and *deviceIndexInHW*. Added description of dSUID enumeration. instead |
| **2017-05-10** | Version 1.0.2 Added outputChannelDescriptions with index-based names |