ZIGBEE DEVICE PHYSICAL INPUT CONFIGURATIONS

INTEGRATOR'S GUIDE

ubisys®

Table of Contents

1. Overview

This document provides guidelines and reference configurations for the physical inputs of ubisys ZigBee products, such like C4, D1(-R), S1(-R), S2(-R), J1(-R) and LD6. These are the same configurations offered via the ubisys Smart Home app for iOS and Android.

This document contains technical material. The reader is assumed to be familiar with the IEEE 802.15.4 standard, the ZigBee Application Foundation including the ZigBee Cluster Library and related technologies and specifications. The intended audience mainly comprises system integrators, who want to take advantage of configurable physical inputs on ZigBee gateways/hubs other than those offered by ubisys.

If you have any questions or need additional support, please visit the Engineering support pages at http://www.ubisys.de/en/smarthome/support.html for contact details.

2. Introduction

Ubisys ZigBee products, which incorporate physical input contacts, provide a wide range of configuration options for these inputs. For example, a mechanical switch with two stable positions can be associated with a single input, configured to send a switch-on command in one position and a switch-off command in the other. Alternatively, a toggle command can be activated with each change in position. Further customization options include setting momentary (push-button) switches to transmit a toggle command with a short press, dimming-up and dimming-down commands with a long press, and so on. This adaptability extends to color light control, time-limited switch-on, enabling/disabling event-driven automated regulation, and similar applications. Essentially, Ubisys switches have the capability to emit nearly any ZigBee command.

3. Recipes

3.1. General Overview

Prior to proceeding, acquaint yourself with the ubisys device setup cluster supported on the ubisys products. You can find a comprehensive description of this cluster in the technical reference manuals for ubisys C4, D1(-R), S1(-R), S2(-R), J1(-R), and LD6. In essence, this cluster facilitates the configuration of binary inputs, establishing mappings for state transitions such as short button press to corresponding over-the-air application layer commands like "toggle." Its flexibility allows for versatile usage patterns, enabling the assignment of one or more physical input pins to each usage pattern.

For example, it is possible to configure a dimmer switch using a single button, where the switch sends “toggle” commands on short press (less than a second) and alternatingly “move up” and “move down” commands on long press. It is also possible to allocate two buttons for similar functionality, such that one button is instructed to send “on” (short press) and “move up” (long press) and the other is instructed to send “off” (short press) and “move down” (long press) commands.

This is done in a way that allows third party commissioning tools and gateways, which are not aware of this advanced manufacturer-specific capability of ubisys products to use standard approaches for provisioning (binding) target devices, for instance using “finding & binding”. Therefore, a number of zigbee endpoints are allocated as logical control units to host outbound clusters for on/off, level control, color control, window covering and scene functionality. Mapping of physical binary inputs to logical application endpoints is also taken care of by the device setup cluster.

This also allows for parameters, for instance move rates, specific target levels, time spans for automatically turning off to be embedded into the command templates. It is also easily extensible. Sending new commands or supporting new clusters does not even require a firmware upgrade [1] .

This universal approach also means that in addition to the configuration of the usage pattern (if the default does not suite the use case), it is also required to create bindings to individual devices, groups of devices or both. Notice you can create multiple bindings per endpoint to control multiple devices or groups at once.

3.1.1. User Interface

The usage patterns presented in this document are supported by the ubisys Smart Home App for iOS and Android, where they can be easily selected using a graphical user interface. If you want to maintain interoperability with these apps, it is recommended that you offer the same functionality to customers in your ecosystem [2]. Examples of how this user interface looks like are provided in the Smart Home App user’s guide.

3.2. Single Stationary Switch as On/Off Switch

This recipe sends an On/Off cluster “on” command when the switch is flipped to its first stable position (e.g. bottom) and an On/Off cluster “off” command when the switch is flipped to its second stable position. It is particularly useful to control multiple lights or, generally speaking, on/off actuators. The advantage in such a scenario is that the lights are inherently synchronized to the same final state, even if they initially had different states, i.e. the lights will be all on or all off after flipping the switch. One drawback is that the lights might already have been turned off via other means (smartphone app, occupancy sensor, pre-determined schedule etc.) and this state is not reflected at the switch. In this case, if the switch is still in the “on” position, but the lights have been turned off, and the user flips the

button, nothing happens, because the lights are already off. Only a subsequent flip results in all lights being turned on.

The idea is that each transition from the released state to the pressed state fires an “on” command and any transition to the released state (regardless whether the transition started from the pressed or kept-pressed state) fires an “off” command.

3.2.1. Example

This would be a configuration for C4, which assigns each of the four inputs as a stationary switch (two stable positions) to a corresponding on/off cluster instance on the primary, secondary, tertiary, and quaternary On/Off control switch endpoints. The shown configuration must be written to the InputActions attribute:

```
41          element type: 0x41 (raw data)
08 00       element count: 0x0008 (8 entries)

06          element #1: six bytes
00          InputAndOptions: 0x00
0D          Transition: released -> pressed
01          Source: Endpoint #1 (hosts the primary on/off client cluster
            on C4)
06 00       Cluster ID: 0x0006 - on/off
01          ZCL Command Template: On

06          element #2: six bytes
00          InputAndOptions: 0x00
03          Transition: any -> released
01          Source: Endpoint #1 (hosts the primary on/off client cluster
            on C4)
06 00       Cluster ID: 0x0006 - on/off
00          ZCL Command Template: Off

06          element #3: six bytes
01          InputAndOptions: 0x01
0D          Transition: released -> pressed
02          Source: Endpoint #2 (hosts the secondary on/off client cluster
            on C4)
06 00       Cluster ID: 0x0006 - on/off
01          ZCL Command Template: On

06          element #4: six bytes
01          InputAndOptions: 0x01
03          Transition: any -> released
02          Source: Endpoint #2 (hosts the secondary on/off client cluster
            on C4)
06 00       Cluster ID: 0x0006 - on/off
00          ZCL Command Template: Off

06          element #5: six bytes
02          InputAndOptions: 0x02
0D          Transition: released -> pressed
03          Source: Endpoint #3 (hosts the tertiary on/off client cluster
            on C4)
06 00       Cluster ID: 0x0006 - on/off
01          ZCL Command Template: On

```

```
44 06          element #6: six bytes
45 02          InputAndOptions: 0x02
46 03          Transition: any -> released
47 03          Source: Endpoint #3 (hosts the tertiary on/off client cluster
48             on C4)
49 06 00       Cluster ID: 0x0006 - on/off
50 00          ZCL Command Template: Off
51
52 06          element #7: six bytes
53 03          InputAndOptions: 0x03
54 0D          Transition: released -> pressed
55 04          Source: Endpoint #4 (hosts quaternary on/off client cluster on C4)
56 06 00       Cluster ID: 0x0006 - on/off
57 01          ZCL Command Template: On
58
59 06          element #8: six bytes
60 03          InputAndOptions: 0x03
61 03          Transition: any -> released
62 04          Source: Endpoint #4 (hosts quaternary on/off client cluster on C4)
63 06 00       Cluster ID: 0x0006 - on/off
64 00          ZCL Command Template: Off
```

3.3. Single Stationary Switch as Toggle Switch

This is similar to the push button version, but intended for switches with two stable positions. Sends a “toggle” command on every flip. It is useful when either one or multiple switches control a single light or, generally speaking, on/off actuator. It is less useful to control a group of lights or multiple actuators, because there is a potential that part of the group is turned on, part is turned off and in such a situation a toggle switch would never be able to turn all targets on or off. For a single target, the advantage is that each switch actuation results in a state change of the target.

The idea is that each transition from the released state to the pressed state fires a “toggle” command and any transition to the released state (regardless whether the transition started from the pressed or kept-pressed state) also fires a “toggle” command.

3.3.1. Example 1

This would be a configuration for C4, which assigns each of the four inputs as a stationary switch (two stable positions) to a corresponding on/off cluster instance on the primary, secondary, tertiary, and quaternary On/Off control switch endpoints. The shown configuration must be written to the InputActions attribute:

```
 1 41          element type: 0x41 (raw data)
 2 08 00       element count: 0x0008 (8 entries)
 3
 4 06          element #1: six bytes
 5 00          InputAndOptions: 0x00
 6 0D          Transition: released -> pressed
 7 01          Source: Endpoint #1 (hosts the primary on/off client cluster
 8             on C4)
 9 06 00       Cluster ID: 0x0006 - on/off
10 02          ZCL Command Template: Toggle
11
12 06          element #2: six bytes
13 00          InputAndOptions: 0x00
```

```
03          Transition: any -> released
01          Source: Endpoint #1 (hosts the primary on/off client cluster
            on C4)
06 00       Cluster ID: 0x0006 - on/off
02          ZCL Command Template: Toggle

06          element #3: six bytes
01          InputAndOptions: 0x01
0D          Transition: released -> pressed
02          Source: Endpoint #2 (hosts the secondary on/off client cluster
            on C4)
06 00       Cluster ID: 0x0006 - on/off
02          ZCL Command Template: Toggle

06          element #4: six bytes
01          InputAndOptions: 0x01
03          Transition: any -> released
02          Source: Endpoint #2 (hosts the secondary on/off client cluster
            on C4)
06 00       Cluster ID: 0x0006 - on/off
02          ZCL Command Template: Toggle

06          element #5: six bytes
02          InputAndOptions: 0x02
0D          Transition: released -> pressed
03          Source: Endpoint #3 (hosts the tertiary on/off client cluster
            on C4)
06 00       Cluster ID: 0x0006 - on/off
02          ZCL Command Template: Toggle

06          element #6: six bytes
02          InputAndOptions: 0x02
03          Transition: any -> released
03          Source: Endpoint #3 (hosts the tertiary on/off client cluster
            on C4)
06 00       Cluster ID: 0x0006 - on/off
02          ZCL Command Template: Toggle

06          element #7: six bytes
03          InputAndOptions: 0x03
0D          Transition: released -> pressed
04          Source: Endpoint #4 (hosts quaternary on/off client cluster on C4)
06 00       Cluster ID: 0x0006 - on/off
02          ZCL Command Template: Toggle

06          element #8: six bytes
03          InputAndOptions: 0x03
03          Transition: any -> released
04          Source: Endpoint #4 (hosts quaternary on/off client cluster on C4)
06 00       Cluster ID: 0x0006 - on/off
02          ZCL Command Template: Toggle
```

3.3.2. Example 2

This is the default configuration for S1, which is aimed at rocker switches (stationary, two stable positions):

```
41          element type: 0x41 (raw data)
02 00       element count: 0x0002 (2 entries)

06          element #1: six bytes
00          InputAndOptions: 0x00
0D          Transition: released -> pressed
02          Source: Endpoint #2 (hosts on/off client cluster on S1)
06 00       Cluster ID: 0x0006 - on/off
02          ZCL Command Template: Toggle

06          element #2: six bytes
00          InputAndOptions: 0x00
03          Transition: any state -> released
02          Source: Endpoint #2 (hosts on/off client cluster on S1)
06 00       Cluster ID: 0x0006 - on/off
02          ZCL Command Template: Toggle
```

3.4. Single Momentary Switch (Push Button) as On/Off Switch

This straightforward recipe alternately transmits an "On" or "Off" command when the momentary switch is pressed. It proves particularly handy for managing multiple lights, or generally speaking, on/off actuators. The benefit in this context lies in the inherent synchronization of the lights to the same ultimate state, even if they may initially have distinctive states. Essentially, all lights will be either on or off after pressing the momentary switch. However, a drawback arises when the lights may have been set to an on/off state through other methods (such as a smartphone app, occupancy sensor, or predetermined schedule) that is not reflected on the momentary switch. For instance, the switch might be set to send the "Off" command on the next press, but the lights have already been turned off. In such cases, pressing the button yields no immediate effect, as the lights are already off. Only a subsequent press results in all lights being turned on.

The concept involves triggering either an "on" command or an "off" command in an alternating manner with each transition from the released state to the pressed state.

3.4.1. Example

This would be a configuration for LD6, which assigns each of the three inputs as a Single Momentary Switch to a corresponding on/off cluster instance on the primary, secondary, tertiary, and quaternary On/Off client cluster endpoints. The shown configuration must be written to the InputActions attribute:

```
41          element type: 0x41 (raw data)
06 00       element count: 0x0006 (6 entries)

06          element #1: six bytes
00          InputAndOptions: 0x00 (the first physical input)
8D          Transition: released -> pressed, has alternate
02          Source: Endpoint #2 (hosts the primary on/off client cluster
            on LD6)
06 00       Cluster ID: 0x0006 - on/off
01          ZCL Command Template: On

06          element #2: six bytes
00          InputAndOptions: 0x00 (the first physical input)
CD          Transition: released -> pressed, is alternate
02          Source: Endpoint #2 (hosts the primary on/off client cluster
            on LD6)
06 00       Cluster ID: 0x0006 - on/off
00          ZCL Command Template: Off

06          element #3: six bytes
01          InputAndOptions: 0x01 (the second physical input)
8D          Transition: released -> pressed, has alternate
03          Source: Endpoint #3 (hosts the secondary on/off client cluster
            on LD6)
06 00       Cluster ID: 0x0006 - on/off
01          ZCL Command Template: On

06          element #4: six bytes
01          InputAndOptions: 0x01 (the second physical input)
CD          Transition: released -> pressed, is alternate
03          Source: Endpoint #3 (hosts the secondary on/off client cluster
            on LD6)
06 00       Cluster ID: 0x0006 - on/off
00          ZCL Command Template: Off

06          element #5: six bytes
02          InputAndOptions: 0x02 (the third physical input)
8D          Transition: released -> pressed, has alternate
04          Source: Endpoint #4 (hosts the tertiary on/off client cluster
            on LD6)
06 00       Cluster ID: 0x0006 - on/off
01          ZCL Command Template: On

06          element #6: six bytes
02          InputAndOptions: 0x02 (the third physical input)
CD          Transition: released -> pressed, is alternate
04          Source: Endpoint #4 (hosts the tertiary on/off client cluster
            on LD6)
06 00       Cluster ID: 0x0006 - on/off
00          ZCL Command Template: Off
```

3.5. Single Momentary Switch (Push Button) as Toggle Switch

This is a very simple recipe, resulting in a “toggle” command being sent on every flip. It is useful when either one or multiple switches control a single light or, generally speaking, on/off actuator. It is less useful to control a group of lights or multiple actuators, because there is a potential that part of the

group is turned on, part is turned off and in such a situation a toggle switch would never be able to turn all targets on or off. For a single target, the advantage is that each switch actuation results in a state change of the target.

The idea is that each transition from the released state to the pressed state fires a “toggle” command.

3.5.1. Example 1

The first example here would be a configuration for C4, which assigns each of the four inputs as a Single Momentary Switch to a corresponding on/off cluster instance on the primary, secondary, tertiary, and quaternary On/Off client cluster endpoints. The shown configuration must be written to the InputActions attribute:

```
41          element type: 0x41 (raw data)
04 00       element count: 0x0004 (4 entries)

06          element #1: six bytes
00          InputAndOptions: 0x00
0D          Transition: released -> pressed
01          Source: Endpoint #1 (hosts the primary on/off client cluster
            on C4)
06 00       Cluster ID: 0x0006 - on/off
02          ZCL Command Template: Toggle

06          element #2: six bytes
01          InputAndOptions: 0x01
0D          Transition: released -> pressed
02          Source: Endpoint #2 (hosts the secondary on/off client cluster
            on C4)
06 00       Cluster ID: 0x0006 - on/off
02          ZCL Command Template: Toggle

06          element #3: six bytes
02          InputAndOptions: 0x02
0D          Transition: released -> pressed
03          Source: Endpoint #3 (hosts the tertiary on/off client cluster
            on C4)
06 00       Cluster ID: 0x0006 - on/off
02          ZCL Command Template: Toggle

06          element #4: six bytes
03          InputAndOptions: 0x03
0D          Transition: released -> pressed
04          Source: Endpoint #4 (hosts quaternary on/off client cluster on C4)
06 00       Cluster ID: 0x0006 - on/off
02          ZCL Command Template: Toggle
```

3.5.2. Example 2

The second example here shows a configuration for C4, in which each of the four inputs is as *Single Stationary Switch as On/off Switch*, *Single Stationary Switch as Toggle Switch*, *Single Momentary Switch (Push Button) as On/Off Switch* and *Single Momentary Switch (Push Button) as Toggle Switch* assigned, to a corresponding on/off cluster instance on the primary, secondary, tertiary, and quaternary On/Off control switch endpoints respectively. The shown configuration must be written to the InputActions attribute:

```
41          element type: 0x41 (raw data)
07 00       element count: 0x0007 (7 entries)

06          element #1: six bytes
00          InputAndOptions: 0x00
0D          Transition: released -> pressed
01          Source: Endpoint #1 (hosts the primary on/off client cluster
            on C4)
06 00       Cluster ID: 0x0006 - on/off
01          ZCL Command Template: On

06          element #2: six bytes
00          InputAndOptions: 0x00
03          Transition: any -> released
01          Source: Endpoint #1 (hosts the primary on/off client cluster
            on C4)
06 00       Cluster ID: 0x0006 - on/off
00          ZCL Command Template: Off

06          element #3: six bytes
01          InputAndOptions: 0x01
0D          Transition: released -> pressed
02          Source: Endpoint #2 (hosts the secondary on/off client cluster
            on C4)
06 00       Cluster ID: 0x0006 - on/off
02          ZCL Command Template: Toggle

06          element #4: six bytes
01          InputAndOptions: 0x01
03          Transition: any -> released
02          Source: Endpoint #2 (hosts the secondary on/off client cluster
            on C4)
06 00       Cluster ID: 0x0006 - on/off
02          ZCL Command Template: Toggle

06          element #5: six bytes
02          InputAndOptions: 0x02 (the third physical input)
8D          Transition: released -> pressed, has alternate
03          Source: Endpoint #3 (hosts the primary on/off client cluster
            on C4)
06 00       Cluster ID: 0x0006 - on/off
01          ZCL Command Template: On

06          element #6: six bytes
02          InputAndOptions: 0x02 (the third physical input)
CD          Transition: released -> pressed, is alternate
03          Source: Endpoint #3 (hosts the secondary on/off client cluster
            on C4)
06 00       Cluster ID: 0x0006 - on/off
00          ZCL Command Template: Off

06          element #7: six bytes
03          InputAndOptions: 0x03
0D          Transition: released -> pressed
04          Source: Endpoint #4 (hosts quaternary on/off client cluster on C4)
06 00       Cluster ID: 0x0006 - on/off
02          ZCL Command Template: Toggle
```

The configurations are for two switches and two buttons, i.e. it configures four inputs available on the device:

```
Element #1 - #2          Single Stationary Switch as On/Off Switch
Element #3 - #4          Single Stationary Switch as Toggle Switch
Element #5 - #6          Single Momentary Switch (Push Button) as On/Off Switch
Element #7               Single Momentary Switch (Push Button) as Toggle
```

3.6. Single Momentary Switch as Timed On/Off Switch

This recipe sends an “on with timed off” command when the push button is pressed. It is particularly useful to control lights in a corridor application, or interfacing with garage door openers, valves, and other equipment, which must return to the off state after a predetermine time. For certain critical applications, this also ensures that the device turns off, unless it is frequently triggered to either turn on, or remain on and thus tolerates missing an off command for equipment that could potentially be damaged or cause damage when operated continuously.

The idea is that each transition from the released state to the pressed state fires an “on with timed off” command; no other transitions are taken into account.

3.6.1. Example 1

This would be a configuration for C4, which assigns each of the four inputs as a momentary switch (one stable position) to a corresponding on/off cluster instance on the primary, secondary, tertiary, and quaternary level control switch endpoints. The shown configuration must be written to the InputActions attribute:

```
41          element type: 0x41 (raw data)
04 00       element count: 0x0004 (4 entries)

0B          element #1: 11 bytes
00          InputAndOptions: 0x00
0D          Transition: released -> pressed
01          Source: Endpoint #1 (hosts the primary on/off client cluster
            on C4)
06 00       Cluster ID: 0x0006 - on/off
42          ZCL Command Template: On with timed off
00          ZCL Command Template: On/off Control field = 0
B0 04       ZCL Command Template: On Time field = 0x04b0 = 1200 * 0.1s = 120s
00 00       ZCL Command Template: Off Wait Time = 0

0B          element #2: 11 bytes
00          InputAndOptions: 0x00
0D          Transition: released -> pressed
02          Source: Endpoint #2 (hosts the primary on/off client cluster
            on C4)
06 00       Cluster ID: 0x0006 - on/off
42          ZCL Command Template: On with timed off
00          ZCL Command Template: On/off Control field = 0
B0 04       ZCL Command Template: On Time field = 0x04b0 = 1200 * 0.1s = 120s
00 00       ZCL Command Template: Off Wait Time = 0

0B          element #3: 11 bytes
00          InputAndOptions: 0x00
0D          Transition: released -> pressed
03          Source: Endpoint #3 (hosts the primary on/off client cluster
            on C4)
06 00       Cluster ID: 0x0006 - on/off
42          ZCL Command Template: On with timed off
00          ZCL Command Template: On/off Control field = 0
B0 04       ZCL Command Template: On Time field = 0x04b0 = 1200 * 0.1s = 120s
00 00       ZCL Command Template: Off Wait Time = 0

0B          element #4: 11 bytes
00          InputAndOptions: 0x00
0D          Transition: released -> pressed
04          Source: Endpoint #4 (hosts the primary on/off client cluster
            on C4)
06 00       Cluster ID: 0x0006 - on/off
42          ZCL Command Template: On with timed off
00          ZCL Command Template: On/off Control field = 0
B0 04       ZCL Command Template: On Time field = 0x04b0 = 1200 * 0.1s = 120s
00 00       ZCL Command Template: Off Wait Time = 0
```

3.6.2. Example 2

This would be a configuration for S1, which assigns its input for a momentary switch (one stable position) to a corresponding on/off cluster instance on the primary on/off switch endpoint. The shown configuration must be written to the InputActions attribute:

```
41          element type: 0x41 (raw data)
01 00       element count: 0x0001 (1 entries)

0B          element #1: 11 bytes
00          InputAndOptions: 0x00
0D          Transition: released -> pressed
02          Source: Endpoint #2 (hosts the primary on/off client cluster
            on C4)
06 00       Cluster ID: 0x0006 - on/off
42          ZCL Command Template: On with timed off
00          ZCL Command Template: On/off Control field = 0
B0 04       ZCL Command Template: On Time field = 0x04b0 = 1200 * 0.1s = 120s
00 00       ZCL Command Template: Off Wait Time = 0
```

For instance, the following facility-app command could be used to write the input configuration above to a particular S1 device, where <device-id> is the identifier of the device as shown in the inventory, and 0x48 is the ZCL data type “array” followed by the actual attribute value above:

```
zcl write <device-id> 232 0xfc00 0x0001 0x48 0x41 0x01 0x00 0x0b 0x00 0x0d 0x02
0x06 0x00 0x42 0x00 0xb0 0x04 0x00 0x00
```

3.7. Single Momentary Switch (Push Button) as Dimmer Switch

This is a more complex recipe designed for dimming a light, or generally speaking adjusting the level (intensity, speed, brightness, …) of a level controllable output. Users are able to turn a light on or off with a short press, i.e. tapping the button for less than a second; and also dim up and down with a long press, i.e. keeping the button pressed for more than a second.

The idea is that each transition from the pressed state to the released state fires a “toggle” command, and each transition from the pressed to the kept-pressed state fires a “move with on/off” command, where the direction is alternating on each subsequent action between “up” and “down”. Notice that the command templates for moving also contain a configurable move rate. Finally, a transition from the kept-pressed to the released state fires a “stop” command.

3.7.1. Example

This the complete default configuration for D1 and D1-R, which is aimed at single push-button operation (momentary, one stable position):

```
41          element type: 0x41 (raw data)
08 00       element count: 0x0008 (8 entries)

06          element #1: six bytes
00          InputAndOptions: 0x00
07          Transition: pressed -> released
02          Source: Endpoint #2 (hosts the primary on/off client cluster
            on D1)
06 00       Cluster ID: 0x0006 - on/off
02          ZCL Command Template: Toggle

08          element #2: eight bytes
00          InputAndOptions: 0x00
```

```
86          Transition: pressed -> kept pressed, has alternate
02          Source: Endpoint #2 (hosts the primary level control client
            cluster on D1)
08 00       Cluster ID: 0x0008 - level control
05 00 32    ZCL Command Template: Move with on/off, upwards, rate = 50

08          element #3: eight bytes
00          InputAndOptions: 0x00
C6          Transition: pressed -> kept pressed, is alternate
02          Source: Endpoint #2 (hosts the primary level control client
            cluster on D1)
08 00       Cluster ID: 0x0008 - level control
05 01 32    ZCL Command Template: Move with on/off, downwards, rate = 50

06          element #4: six bytes
00          InputAndOptions: 0x00
0B          Transition: kept pressed -> released
02          Source: Endpoint #2 (hosts the primary level control client
            cluster on D1)
08 00       Cluster ID: 0x0008 - level control
07          ZCL Command Template: Stop with on/off

06          element #5: six bytes
01          InputAndOptions: 0x01
07          Transition: pressed -> released
03          Source: Endpoint #3 (hosts the secondary on/off client cluster
            on D1)
06 00       Cluster ID: 0x0006 - on/off
02          ZCL Command Template: Toggle

08          element #6: eight bytes
01          InputAndOptions: 0x01
86          Transition: pressed -> kept pressed, has alternate
03          Source: Endpoint #3 (hosts the secondary level control client
            cluster on D1)
08 00       Cluster ID: 0x0008 - level control
05 00 32    ZCL Command Template: Move with on/off, upwards, rate = 50

08          element #7: eight bytes
01          InputAndOptions: 0x01
C6          Transition: pressed -> kept pressed, is alternate
03          Source: Endpoint #3 (hosts the secondary level control client
            cluster on D1)
08 00       Cluster ID: 0x0008 - level control
05 01 32    ZCL Command Template: Move with on/off, downwards, rate = 50

06          element #8: six bytes
01          InputAndOptions: 0x01
0B          Transition: kept pressed -> released
03          Source: Endpoint #3 (hosts the secondary level control client cluster
            on D1)
08 00       Cluster ID: 0x0008 - level control
07          ZCL Command Template: Stop with on/off
```

This will allow to control a dimmer with one push button. A short press will toggle the light on/off, while a longer press starts dimming up or down (alternating) in order to allow adjusting the brightness with the button. Dimming stops, when the button is released. The code is for two buttons, i.e. it configures both inputs available on the device.

3.8. Double Momentary Switch (Push Buttons) as Dimmer Switch

This is a more complex recipe designed for dimming a light, or generally speaking adjusting the level (intensity, speed, brightness, …) of a level controllable output utilizing two momentary switches (push buttons, one stable position). Users are able to turn a light on with a short press on one button (e.g. left button of a double switch), i.e. tapping the button for less than a second; and also dim up with a long press on the same button, i.e. keeping the button pressed for more than a second; the second button is used for the opposite actions, i.e. a short press on the second (e.g. right button of a double switch) will turn the lights off; a long press on this button will dim the lights down.

The idea is that each transition from the pressed state to the released state fires an “on” command for the first button (“off” command for the second button), and each transition from the pressed to the kept-pressed state fires a “move with on/off” command, where the direction is “up” for the first button (“down” for the second button). Notice that the command templates for moving also contain a configurable move rate. Finally, a transition from the kept-pressed to the released state fires a “stop” command (same for both buttons).

3.8.1. Example

The following example shows the input action micro-code for using two push-buttons to control a target dimmer (whether it be the local output or a remote device) in an up/down manner, i.e. one button is used to turn the light(s) on and dim brighter, the other one to turn the light(s) off and dim darker:

```
41          element type: 0x41 (raw data)
06 00       element count: 0x0006 (6 entries)

06          element #1: six bytes
00          InputAndOptions: 0x00
07          Transition: pressed -> released
02          Source: Endpoint #2 (hosts the primary on/off client cluster
            on D1)
06 00       Cluster ID: 0x0006 - on/off
01          ZCL Command Template: Turn on

08          element #2: eight bytes
00          InputAndOptions: 0x00
06          Transition: pressed -> kept pressed
02          Source: Endpoint #2 (hosts the primary level control client
            cluster on D1)
08 00       Cluster ID: 0x0008 - level control
05 00 32    ZCL Command Template: Move with on/off, upwards, rate = 50

06          element #3: six bytes
00          InputAndOptions: 0x00
0B          Transition: kept pressed -> released
02          Source: Endpoint #2 (hosts the secondary level control client
            cluster on D1)
08 00       Cluster ID: 0x0008 - level control
07          ZCL Command Template: Stop with on/off

06          element #4: six bytes
01          InputAndOptions: 0x01
07          Transition: pressed -> released
02          Source: Endpoint #2 (hosts the primary on/off client cluster
            on D1)
06 00       Cluster ID: 0x0006 - on/off
00          ZCL Command Template: Turn off

08          element #5: eight bytes
01          InputAndOptions: 0x01
06          Transition: pressed -> kept pressed
02          Source: Endpoint #2 (hosts the primary level control client
            cluster on D1)
08 00       Cluster ID: 0x0008 - level control
05 01 32    ZCL Command Template: Move with on/off, downwards, rate = 50

06          element #6: six bytes
01          InputAndOptions: 0x01
0B          Transition: kept pressed -> released
02          Source: Endpoint #2 (hosts the secondary level control client
            cluster on D1)
08 00       Cluster ID: 0x0008 - level control
07          ZCL Command Template: Stop with on/off
```

3.9. Double Momentary Switch (Push Buttons) as Shutter Switch

This is a fairly simple recipe designed for controlling a window blind utilizing two momentary switches (push buttons, one stable position). Users are able to adjust the tilt angle using short presses and have the shutter drive to its upper and lower bounds using a long press. One button is used for the

up direction, another one for the down direction.

The idea is that each transition from the released state to the pressed state fires a “move up/open” command for the first button (“move down/close” command for the second button), and each transition from the pressed to the released state fires a “stop” command (same for both buttons). The trick here is to don’t send a stop command when the transition originates in a kept-pressed command.

3.9.1. Example

This is the default configuration for J1 and J1-R, which is aimed at dual push-button operation (momentary, one stable position):

```
41          element type: 0x41 (raw data)
04 00       element count: 0x0004 (4 entries)

06          element #1: six bytes
00          InputAndOptions: 0x00
0D          Transition: released -> pressed
02          Source: Endpoint #2 (hosts window covering client cluster on J1)
02 01       Cluster ID: 0x0102 - window covering
00          ZCL Command Template: Move up/open

06          element #2: six bytes
00          InputAndOptions: 0x00
07          Transition: pressed -> released
02          Source: Endpoint #2 (hosts window covering client cluster on J1)
02 01       Cluster ID: 0x0102 - window covering
02          ZCL Command Template: Stop

06          element #3: six bytes
01          InputAndOptions: 0x01
0D          Transition: released -> pressed
02          Source: Endpoint #2 (hosts window covering client cluster on J1)
02 01       Cluster ID: 0x0102 - window covering
01          ZCL Command Template: Move down/close

06          element #4: six bytes
01          InputAndOptions: 0x01
07          Transition: pressed -> released
02          Source: Endpoint #2 (hosts window covering client cluster on J1)
02 01       Cluster ID: 0x0102 - window covering
02          ZCL Command Template: Stop
```

A short press will move up/down and stop when released, while a long press will move up/down without stopping before the fully open or fully closed position is reached, respectively. This is particularly useful for lift & tilt blinds, but also generally suitable for all kinds of attached devices.

3.10. Double Stationary Switch as Shutter Switch

This is a fairly simple recipe designed for controlling a window blind utilizing two momentary switches (push buttons, one stable position). Users are able to adjust the tilt angle using short presses and have the shutter drive to its upper and lower bounds using a long press. One button is used for the up direction, another one for the down direction.

The idea is that each transition from the released state to the pressed state fires a “move up/open” command for the first button (“move down/close” command for the second button), and each transition from the pressed to the released state fires a “stop” command (same for both buttons). The trick here is to not send a stop command when the transition originates in a kept-pressed command.

3.10.1. Example

If stationary switches are connected to the inputs of a J1 or J1-R the following instructions shall be used:

```
41          element type: 0x41 (raw data)
04 00       element count: 0x0004 (4 entries)

06          element #1: six bytes
00          InputAndOptions: 0x00
0D          Transition: released -> pressed
02          Source: Endpoint #2 (hosts window covering client cluster on J1)
02 01       Cluster ID: 0x0102 - window covering
00          ZCL Command Template: Move up/open

06          element #2: six bytes
00          InputAndOptions: 0x00
03          Transition: any state -> released
02          Source: Endpoint #2 (hosts window covering client cluster on J1)
02 01       Cluster ID: 0x0102 - window covering
02          ZCL Command Template: Stop

06          element #3: six bytes
01          InputAndOptions: 0x01
0D          Transition: released -> pressed
02          Source: Endpoint #2 (hosts window covering client cluster on J1)
02 01       Cluster ID: 0x0102 - window covering
01          ZCL Command Template: Move down/close

06          element #4: six bytes
01          InputAndOptions: 0x01
03          Transition: any state -> released
02          Source: Endpoint #2 (hosts window covering client cluster on J1)
02 01       Cluster ID: 0x0102 - window covering
02          ZCL Command Template: Stop
```

Here, the blind moves as long as either switch is turned on. As soon as it is turned off, motion stops. The same approach can be applied to a C4, which allows two connect four switches and thus provides two window covering controllers (or a mix of one window covering controller plus one or two other functions).

3.11. Single Switch (Push Button) as Scene Selector Switch

This recipe allows to recall one or two scenes, i.e. one for a short press and optionally one for a long press. In contrast to all other input actions, a recall scene command will always be sent as group cast to the group specified in the payload of the recall scene command.

The idea is that each transition from the pressed state to the released state fires a “recall scene” command for one scene and a transition to the pressed state to the kept-pressed state fires a “recall scene” command for another scene.

3.11.1. Example

This would be a configuration for C4, which assigns each of the four inputs as a stationary switch (two stable positions) to a corresponding scene cluster instance on the primary, secondary, tertiary, and quaternary level control switch endpoints. Each switch can recall two separate scenes (they need not address the same group). Contrary to all other examples, no binding is required here on the source endpoint to targets. This is to allow mixing groups and making sure the group address for the multicast matches the group in the payload. The shown configuration must be written to the InputActions attribute:

```
41          element type: 0x41 (raw data)
08 00       element count: 0x0008 (8 entries)

06          element #1: six bytes
00          InputAndOptions: 0x00
07          Transition: pressed -> released
01          Source: Endpoint #1 (hosts the primary scene client cluster on C4)
05 00       Cluster ID: 0x0005 - scenes
05          ZCL Command Template: Recall scene,
34 12                             group ID = 0x1234,
56                                scene ID = 0x56

06          element #2: six bytes
00          InputAndOptions: 0x00
06          Transition: pressed -> kept-pressed
01          Source: Endpoint #1 (hosts the primary scene client cluster on C4)
05 00       Cluster ID: 0x0005 - scenes
05          ZCL Command Template: Recall scene,
9a 78                             group ID = 0x789a,
bc                                scene ID = 0xbc

06          element #3: six bytes
01          InputAndOptions: 0x01
07          Transition: pressed -> released
02          Source: Endpoint #2 (hosts the secondary scene client cluster
            on C4)
05 00       Cluster ID: 0x0005 - scenes
05          ZCL Command Template: Recall scene,
22 11                             group ID = 0x1122,
00                                scene ID = 0x00

06          element #4: six bytes
01          InputAndOptions: 0x01
06          Transition: pressed -> kept-pressed
02          Source: Endpoint #2 (hosts the secondary scene client cluster
            on C4)
05 00       Cluster ID: 0x0005 - scenes
05          ZCL Command Template: Recall scene,
22 11                             group ID = 0x1122,
01                                scene ID = 0x01

06          element #5: six bytes
02          InputAndOptions: 0x02
07          Transition: pressed -> released
03          Source: Endpoint #3 (hosts the tertiary scene client cluster
            on C4)
05 00       Cluster ID: 0x0005 - scenes
```

```
05          ZCL Command Template: Recall scene,
44 33                             group ID = 0x3344,
55                                scene ID = 0x55

06          element #6: six bytes
02          InputAndOptions: 0x02
06          Transition: pressed -> kept-pressed
03          Source: Endpoint #3 (hosts the tertiary scene client cluster
            on C4)
05 00       Cluster ID: 0x0005 - scenes
05          ZCL Command Template: Recall scene,
44 33                             group ID = 0x3344,
22                                scene ID = 0x22

06          element #7: six bytes
03          InputAndOptions: 0x03
07          Transition: pressed -> released
04          Source: Endpoint #4 (hosts quaternary scene client cluster on C4)
05 00       Cluster ID: 0x0005 - scenes
05          ZCL Command Template: Recall scene,
66 55                             group ID = 0x5566,
77                                scene ID = 0x77

06          element #8: six bytes
03          InputAndOptions: 0x03
06          Transition: pressed -> kept-pressed
04          Source: Endpoint #4 (hosts quaternary scene client cluster on C4)
05 00       Cluster ID: 0x0005 - scenes
05          ZCL Command Template: Recall scene,
99 88                             group ID = 0x8899,
aa                                scene ID = 0xaa
```

3.12. Single Stationary Switch as Scene Selector Switch

This recipe allows to recall one or two scenes, i.e. one in the first stable position of the switch and optionally one in the second. In contrast to all other input actions, a recall scene command will always be sent as group cast to the group specified in the payload of the recall scene command.

The idea is that each transition from the released state to the pressed state fires a “recall scene” command for one scene and any transition to the released state (regardless whether the transition started from the pressed or kept-pressed state) fires a “recall scene” command for another scene.

3.12.1. Example

This would be a configuration for C4, which assigns each of the four inputs as a stationary switch (two stable positions) to a corresponding scene cluster instance on the primary, secondary, tertiary, and quaternary level control switch endpoints. Each switch can recall two separate scenes (they need not address the same group). Contrary to all other examples, no binding is required here on the source endpoint to targets. This is to allow mixing groups and making sure the group address for the multicast matches the group in the payload. The shown configuration must be written to the InputActions attribute:

```
41          element type: 0x41 (raw data)
08 00       element count: 0x0008 (8 entries)

```

```
06          element #1: six bytes
00          InputAndOptions: 0x00
0D          Transition: released -> pressed
01          Source: Endpoint #1 (hosts the primary scene client cluster on C4)
05 00       Cluster ID: 0x0005 - scenes
05          ZCL Command Template: Recall scene,
34 12                           group ID = 0x1234,
56                              scene ID = 0x56

06          element #2: six bytes
00          InputAndOptions: 0x00
03          Transition: any -> released
01          Source: Endpoint #1 (hosts the primary scene client cluster on C4)
05 00       Cluster ID: 0x0005 - scenes
05          ZCL Command Template: Recall scene,
9a 78                           group ID = 0x789a,
bc                              scene ID = 0xbc

06          element #3: six bytes
01          InputAndOptions: 0x01
0D          Transition: released -> pressed
02          Source: Endpoint #2 (hosts the secondary scene client cluster
            on C4)
05 00       Cluster ID: 0x0005 - scenes
05          ZCL Command Template: Recall scene,
22 11                           group ID = 0x1122,
00                              scene ID = 0x00

06          element #4: six bytes
01          InputAndOptions: 0x01
03          Transition: any -> released
02          Source: Endpoint #2 (hosts the secondary scene client cluster
            on C4)
05 00       Cluster ID: 0x0005 - scenes
05          ZCL Command Template: Recall scene,
22 11                           group ID = 0x1122,
01                              scene ID = 0x01

06          element #5: six bytes
02          InputAndOptions: 0x02
0D          Transition: released -> pressed
03          Source: Endpoint #3 (hosts the tertiary scene client cluster
            on C4)
05 00       Cluster ID: 0x0005 - scenes
05          ZCL Command Template: Recall scene,
44 33                           group ID = 0x3344,
55                              scene ID = 0x55

06          element #6: six bytes
02          InputAndOptions: 0x02
03          Transition: any -> released
03          Source: Endpoint #3 (hosts the tertiary scene client cluster
            on C4)
05 00       Cluster ID: 0x0005 - scenes
05          ZCL Command Template: Recall scene,
44 33                           group ID = 0x3344,
22                              scene ID = 0x22

06          element #7: six bytes
```

```
03          InputAndOptions: 0x03
0D          Transition: released -> pressed
04          Source: Endpoint #4 (hosts quaternary scene client cluster on C4)
05 00       Cluster ID: 0x0005 - scenes
05          ZCL Command Template: Recall scene,
66 55                             group ID = 0x5566,
77                                scene ID = 0x77

06          element #8: six bytes
03          InputAndOptions: 0x03
03          Transition: any -> released
04          Source: Endpoint #4 (hosts quaternary scene client cluster on C4)
05 00       Cluster ID: 0x0005 - scenes
05          ZCL Command Template: Recall scene,
99 88                             group ID = 0x8899,
aa                                scene ID = 0xaa
```

3.13. Single Momentary Switch (Push Button) as White Tone Button

The concept involves modifying the color temperature using a single momentary switch. Users can either set a tunable-white light to a predefined color temperature by briefly pressing the button (less than a second), or adjust the color temperature up or down at a designated rate by holding the button for an extended period (more than a second).

The idea is that each transition from the pressed state to the released state fires a “Move to Color Temperature” command with specified target color temperature in mireds and transition time, and each transition from the pressed to the kept-pressed state fires a “Move Color Temperature” command, where the mode is alternating on each subsequent action between “Up” and “Down”. Notice that the command templates for moving color temperature also contain a configurable move rate and upper and lower limits of color temperature. Finally, a transition from the kept-pressed to the released state fires a “stop” command, i.e. the process of adjusting the color temperature between cold and warm temperature is stopped.

3.13.1. Example

This would be a default configuration for LD6, which assigns each of the three inputs as a dimmer switch to a corresponding color control cluster instance on the primary, secondary, tertiary, and quaternary color control switch endpoints. The shown configuration must be written to the InputActions attribute:

```
41          element type: 0x41 (raw data)
0C 00       element count: 0x000C (12 entries)

0a          element #1: ten bytes
00          InputAndOptions: 0x00 (the first physical input)
07          Transition: pressed -> released
02          Source: Endpoint #2 (hosts the primary color control client
            cluster on LD6)
00 03       Cluster ID: 0x0300 (Color Control)
0a          ZCL Command Template: Move to Color Temperature,
fa 00                             ColorTemperatureMireds = 250 (4000K),
0a 00                             TransitionTime = 10s

0d          element #2: thirteen bytes
```

```
00          InputAndOptions: 0x00 (the first physical input)
86          Transition: pressed -> kept pressed, has alternate
02          Source: Endpoint #2 (hosts the primary color control client
            cluster on LD6)
00 03       Cluster ID: 0x0300 (Color Control)
4b          ZCL Command Template: Move Color Temperature,
01                          MoveMode = Up
19 00                       MoveRate = 25
00 00                       ColorTemperatureMinimumMireds = 0x0000
00 00                       ColorTemperatureMaximumMireds = 0x0000

0d          element #3: thirteen bytes
00          InputAndOptions: 0x00 (the first physical input)
c6          Transition: pressed -> kept pressed, is alternate
02          Source: Endpoint #2 (hosts the primary color control client
            cluster on LD6)
00 03       Cluster ID: 0x0300 (Color Control)
4b          ZCL Command Template: Move Color Temperature,
03                          MoveMode = Down
19 00                       MoveRate = 25
00 00                       ColorTemperatureMinimumMireds = 0x0000
00 00                       ColorTemperatureMaximumMireds = 0x0000

06          element #4: six bytes
00          InputAndOptions: 0x00 (the first physical input)
0b          Transition: kept pressed -> released
02          Source: Endpoint #2 (hosts the primary color control client
            cluster on LD6)
00 03       Cluster ID: 0x0300 (Color Control)
47          ZCL Command Template: Stop Move Step


0a          element #5: ten bytes
01          InputAndOptions: 0x01 (the second physical input)
07          Transition: pressed -> released
03          Source: Endpoint #3 (hosts the secondary color control client
            cluster on LD6)
00 03       Cluster ID: 0x0300 (Color Control)
0a          ZCL Command Template: Move to Color Temperature,
fa 00                       ColorTemperatureMireds = 250,
0a 00                       TransitionTime = 10s

0d          element #6: thirteen bytes
01          InputAndOptions: 0x01 (the second physical input)
86          Transition: pressed -> kept pressed, has alternate
03          Source: Endpoint #3 (hosts the secondary color control client
            cluster on LD6)
00 03       Cluster ID: 0x0300 (Color Control)
4b          ZCL Command Template: Move Color Temperature,
01                          MoveMode = Up
19 00                       MoveRate = 25
00 00                       ColorTemperatureMinimumMireds = 0x0000
00 00                       ColorTemperatureMaximumMireds = 0x0000

0d          element #7: thirteen bytes
01          InputAndOptions: 0x01 (the second physical input)
c6          Transition: pressed -> kept pressed, is alternate
03          Source: Endpoint #3 (hosts the secondary color control client
            cluster on LD6)
```

```
00 03       Cluster ID: 0x0300 (Color Control)
4b          ZCL Command Template: Move Color Temperature,
03                            MoveMode = Down
19 00                         MoveRate = 25
00 00                         ColorTemperatureMinimumMireds = 0x0000
00 00                         ColorTemperatureMaximumMireds = 0x0000

06          element #8: six bytes
01          InputAndOptions: 0x01 (the second physical input)
0b          Transition: kept pressed -> released
03          Source: Endpoint #3 (hosts the secondary color control client
            cluster on LD6)
00 03       Cluster ID: 0x0300 (Color Control)
47          ZCL Command Template: Stop Move Step


0a          element #9: ten bytes
02          InputAndOptions: 0x02 (the third physical input)
07          Transition: pressed -> released
04          Source: Endpoint #4 (hosts the tertiary color control client
            cluster on LD6)
00 03       Cluster ID: 0x0300 (Color Control)
0a          ZCL Command Template: Move to Color Temperature,
fa 00                         ColorTemperatureMireds = 250,
0a 00                         TransitionTime = 10s

0d          element #10: thirteen bytes
02          InputAndOptions: 0x02 (the third physical input)
86          Transition: pressed -> kept pressed, has alternate
04          Source: Endpoint #4 (hosts the tertiary color control client
            cluster on LD6)
00 03       Cluster ID: 0x0300 (Color Control)
4b          ZCL Command Template: Move Color Temperature,
01                            MoveMode = Up
19 00                         MoveRate = 25
00 00                         ColorTemperatureMinimumMireds = 0x0000
00 00                         ColorTemperatureMaximumMireds = 0x0000

0d          element #11: thirteen bytes
02          InputAndOptions: 0x02 (the third physical input)
c6          Transition: pressed -> kept pressed, is alternate
04          Source: Endpoint #4 (hosts the tertiary color control client
            cluster on LD6)
00 03       Cluster ID: 0x0300 (Color Control)
4b          ZCL Command Template: Move Color Temperature,
03                            MoveMode = Down
19 00                         MoveRate = 25
00 00                         ColorTemperatureMinimumMireds = 0x0000
00 00                         ColorTemperatureMaximumMireds = 0x0000

06          element #12: six bytes
02          InputAndOptions: 0x02 (the third physical input)
0b          Transition: kept pressed -> released
04          Source: Endpoint #4 (hosts the tertiary color control client
            cluster on LD6)
00 03       Cluster ID: 0x0300 (Color Control)
47          ZCL Command Template: Stop Move Step
```

3.14. Single Stationary Switch as Automation Switch

This concept involves generating standard notifications for the alteration of a solitary stationary switch's station. Upon receiving such a notification about a station change, it becomes the responsibility of the recipient to interpret the received station change and execute the required actions accordingly. This method offers significant flexibility and enhances interoperability, as those receiving such notifications typically encapsulate the application logic and are most knowledgeable about appropriate responses when a control switch is flipped.

One specific scenario involves employing generic state change notifications as catalysts for initiating or concluding automated control operations via a ubisys gateway G1. In this context, a standard switch transforms into an automation switch, with the requisite application logic executed on the G1.

When a physical switch is toggled to its initial stable position, an automation switch command is dispatched with the fixed position "1." Conversely, when the switch is flipped to its second stable position, an automation switch command is transmitted with the fixed position "0." Depending on the newly indicated position, automated control activities can be either initiated or halted.

As the frame definition precisely matches the Matter command definition for the commands of the Generic Switch device type, notification frames generated by an Automation switch could be seamlessly tunneled from the Zigbee domain to the Matter domain via a Zigbee/Matter gateway, facilitating smooth integration.

3.14.1. Example

This would be a configuration for LD6, which assigns each of the three inputs as a stationary switch with two station states (start/stop) to a corresponding Managed Input cluster instance on the primary, secondary and tertiary control switch endpoints. The shown configuration must be written to the InputActions attribute:

```
41          element type: 0x41 (raw data)
06 00       element count: 0x0006 (6 entries)

09          element #1: nine bytes
10          InputAndOptions: 0x10 (the first physical input with manufacturer
            specific configurations)
0d          Transition: released -> pressed
02          Source: Endpoint #2 (hosts the primary Managed Input client
            cluster on LD6)
02 fc       Manufacturer Specific Cluster ID: 0xfc02 (Managed Input)
f2 10       Manufacturer code of ubisys: 0x10f2
00          Manufacturer Specific Command Template: Switch latched,
01                                                  NewPosition = 0x01

09          element #2: nine bytes
10          InputAndOptions: 0x10 (the first physical input with manufacturer
            specific configurations)
03          Transition: any -> released
02          Source: Endpoint #2 (hosts the primary Managed Input client
            cluster on LD6)
02 fc       Manufacturer Specific Cluster ID: 0xfc02 (Managed Input)
f2 10       Manufacturer code of ubisys: 0x10f2
00          Manufacturer Specific Command Template: Switch latched,
00                                                  NewPosition = 0x00

```

```
09          element #3: nine bytes
11          InputAndOptions: 0x11 (the second physical input with manufacturer
            specific configurations)
0d          Transition: released -> pressed
03          Source: Endpoint #3 (hosts the secondary Managed Input client
            cluster on LD6)
02 fc       Manufacturer Specific Cluster ID: 0xfc02 (Managed Input)
f2 10       Manufacturer code of ubisys: 0x10f2
00          Manufacturer Specific Command Template: Switch latched,
01                                                  NewPosition = 0x01

09          element #4: nine bytes
11          InputAndOptions: 0x11 (the second physical input with manufacturer
            specific configurations)
03          Transition: any -> released
03          Source: Endpoint #3 (hosts the secondary Managed Input client
            cluster on LD6)
02 fc       Manufacturer Specific Cluster ID: 0xfc02 (Managed Input)
f2 10       Manufacturer code of ubisys: 0x10f2
00          Manufacturer Specific Command Template: Switch latched,
00                                                  NewPosition = 0x00

09          element #5: nine bytes
12          InputAndOptions: 0x12 (the third physical input with manufacturer
            specific configurations)
0d          Transition: released -> pressed
04          Source: Endpoint #4 (hosts the tertiary Managed Input client
            cluster on LD6)
02 fc       Manufacturer Specific Cluster ID: 0xfc02 (Managed Input)
f2 10       Manufacturer code of ubisys: 0x10f2
00          Manufacturer Specific Command Template: Switch latched,
01                                                  NewPosition = 0x01

09          element #6: nine bytes
12          InputAndOptions: 0x12 (the third physical input with manufacturer
            specific configurations)
03          Transition: any -> released
04          Source: Endpoint #4 (hosts the tertiary Managed Input client
            cluster on LD6)
02 fc       Manufacturer Specific Cluster ID: 0xfc02 (Managed Input)
f2 10       Manufacturer code of ubisys: 0x10f2
00          Manufacturer Specific Command Template: Switch latched,
00                                                  NewPosition = 0x00
```

3.15. Single Momentary Switch (Push Button) as Automation Button

This concept involves generating standard notifications for the state changes of a momentary switch. Upon receiving such a notification about a station change, it becomes the responsibility of the recipient to interpret the received station change and execute the required actions accordingly. This method offers significant flexibility and enhances interoperability, as those receiving such notifications typically encapsulate the application logic and are most knowledgeable about appropriate responses when a momentary switch is short-/long-pressed or released.

One specific scenario involves employing generic state change notifications as catalysts for initiating or concluding automated control operations via a ubisys gateway G1. In this context, a standard push button transforms into an automation button, with the requisite application logic executed on the G1.

The idea is that each transition from the released state to the pressed state fires a “Initial Short Press” command, each transition from the pressed to the kept-pressed state fires a “Long Press” command, each transition from the pressed to the released state fires a “Short Release” command, and each transition from the kept-pressed to the released state fires a “Long Release” command. Upon receipt of a “Short Press” or “Long Press” command, automated control activities cab be either paused or initiated.

As the frame definition precisely matches the Matter command definition for the commands of the Generic Switch device type, notification frames generated by an Automation button could be seamlessly tunneled from the Zigbee domain to the Matter domain via a Zigbee/Matter gateway, facilitating smooth integration.

3.15.1. Example

This would be a configuration for LD6, which assigns each of the three inputs as a momentary switch to a corresponding managed input cluster instance on the primary, secondary and tertiary control switch endpoints. The shown configuration must be written to the InputActions attribute:

```
41          element type: 0x41 (raw data)
0c 00       element count: 0x000c (12 entries)

09          element #1: nine bytes
10          InputAndOptions: 0x10 (the first physical input with manufacturer
            specific configurations.)
0d          Transition: released -> pressed
02          Source: Endpoint #2 (hosts the primary Managed Input client
            cluster on LD6)
02 fc       Manufacturer Specific Cluster ID: 0xfc02 (Managed Input)
f2 10       Manufacturer code of ubisys: 0x10f2
01          Manufacturer Specific Command Template: initial short press,
01                                                  currentPosition = 0x01

09          element #2: nine bytes
10          InputAndOptions: 0x10 (the first physical input with manufacturer
            specific configurations.)
06          Transition: pressed -> kept pressed
02          Source: Endpoint #2 (hosts the primary Managed Input client
            cluster on LD6)
02 fc       Manufacturer Specific Cluster ID: 0xfc02 (Managed Input)
f2 10       Manufacturer code of ubisys: 0x10f2
02          Manufacturer Specific Command Template: long press,
01                                                  currentPosition = 0x01

09          element #3: nine bytes
10          InputAndOptions: 0x10 (the first physical input with manufacturer
            specific configurations.)
07          Transition: pressed -> released
02          Source: Endpoint #2 (hosts the primary Managed Input client
            cluster on LD6)
02 fc       Manufacturer Specific Cluster ID: 0xfc02 (Managed Input)
f2 10       Manufacturer code of ubisys: 0x10f2
03          Manufacturer Specific Command Template: short release,
01                                                  previousPosition = 0x01

09          element #4: nine bytes
10          InputAndOptions: 0x10 (the first physical input with manufacturer
```

```
             specific configurations.)
0b           Transition: kept pressed -> released
02           Source: Endpoint #2 (hosts the primary Managed Input client
             cluster on LD6)
02 fc        Manufacturer Specific Cluster ID: 0xfc02 (Managed Input)
f2 10        Manufacturer code of ubisys: 0x10f2
04           Manufacturer Specific Command Template: long release,
01                                                   previousPosition = 0x01

09           element #5: nine bytes
11           InputAndOptions: 0x11 (the second physical input with manufacturer
             specific configurations.)
0d           Transition: released -> pressed
03           Source: Endpoint #3 (hosts the secondary Managed Input client
             cluster on LD6)
02 fc        Manufacturer Specific Cluster ID: 0xfc02 (Managed Input)
f2 10        Manufacturer code of ubisys: 0x10f2
01           Manufacturer Specific Command Template: initial short press,
01                                                   currentPosition = 0x01

09           element #6: nine bytes
11           InputAndOptions: 0x11 (the second physical input with manufacturer
             specific configurations.)
06           Transition: pressed -> kept pressed
03           Source: Endpoint #3 (hosts the secondary Managed Input client
             cluster on LD6)
02 fc        Manufacturer Specific Cluster ID: 0xfc02 (Managed Input)
f2 10        Manufacturer code of ubisys: 0x10f2
02           Manufacturer Specific Command Template: long press,
01                                                   currentPosition = 0x01

09           element #7: nine bytes
11           InputAndOptions: 0x11 (the second physical input with manufacturer
             specific configurations.)
07           Transition: pressed -> released
03           Source: Endpoint #3 (hosts the secondary Managed Input client
             cluster on LD6)
02 fc        Manufacturer Specific Cluster ID: 0xfc02 (Managed Input)
f2 10        Manufacturer code of ubisys: 0x10f2
03           Manufacturer Specific Command Template: short release,
01                                                   previousPosition = 0x01

09           element #8: nine bytes
11           InputAndOptions: 0x11 (the second physical input with manufacturer
             specific configurations.)
0b           Transition: kept pressed -> released
03           Source: Endpoint #3 (hosts the secondary Managed Input client
             cluster on LD6)
02 fc        Manufacturer Specific Cluster ID: 0xfc02 (Managed Input)
f2 10        Manufacturer code of ubisys: 0x10f2
04           Manufacturer Specific Command Template: long release,
01                                                   previousPosition = 0x01

09           element #9: nine bytes
12           InputAndOptions: 0x12 (the third physical input with manufacturer
             specific configurations.)
0d           Transition: released -> pressed
04           Source: Endpoint #4 (hosts the tertiary Managed Input client
             cluster on LD6)
```

```
 98 02 fc       Manufacturer Specific Cluster ID: 0xfc02 (Managed Input)
 99 f2 10       Manufacturer code of ubisys: 0x10f2
100 01          Manufacturer Specific Command Template: initial short press,
101 01                                                  currentPosition = 0x01
102
103 09          element #10: nine bytes
104 12          InputAndOptions: 0x12 (the third physical input with manufacturer
105             specific configurations.)
106 06          Transition: pressed -> kept pressed
107 04          Source: Endpoint #4 (hosts the tertiary Managed Input client
108             cluster on LD6)
109 02 fc       Manufacturer Specific Cluster ID: 0xfc02 (Managed Input)
110 f2 10       Manufacturer code of ubisys: 0x10f2
111 02          Manufacturer Specific Command Template: long press,
112 01                                                  currentPosition = 0x01
113
114 09          element #11: nine bytes
115 12          InputAndOptions: 0x12 (the third physical input with manufacturer
116             specific configurations.)
117 07          Transition: pressed -> released
118 04          Source: Endpoint #4 (hosts the tertiary Managed Input client
119             cluster on LD6)
120 02 fc       Manufacturer Specific Cluster ID: 0xfc02 (Managed Input)
121 f2 10       Manufacturer code of ubisys: 0x10f2
122 03          Manufacturer Specific Command Template: short release,
123 01                                                  previousPosition = 0x01
124
125 09          element #12: nine bytes
126 12          InputAndOptions: 0x12 (the third physical input with manufacturer
127             specific configurations.)
128 0b          Transition: kept pressed -> released
129 04          Source: Endpoint #4 (hosts the tertiary Managed Input client
130             cluster on LD6)
131 02 fc       Manufacturer Specific Cluster ID: 0xfc02 (Managed Input)
132 f2 10       Manufacturer code of ubisys: 0x10f2
133 04          Manufacturer Specific Command Template: long release,
134 01                                                  previousPosition = 0x01
```

[1] Support for new clusters should be accompanied with a firmware upgrade, which adds the specific outbound cluster to the simple descriptor of the logical control unit endpoint, or introduces a new endpoint with the new cluster to facilitate finding & binding. For the raw functionality, it is not strictly required, though.

[2] Qualified customers may license C (at least C11 language support is required) and Java classes from ubisys, which help with identification and assembly of usage patterns.

4. Revision History

Revision	Date	Remarks
0.1	01/24/2013	Initial Version
0.5	12/19/2014	Added advanced support for recall scene command. Recall scene commands are now always sent to the group ID in the payload and don't require an entry in the binding table
0.9	11/25/2014	Preliminary version (for internal use only).
1.0	05/31/2017	Initial Public Version.
2.0	03/01/2021	Added recommended configuration for a push button that sends "on with timed off" commands.
3.0	08/01/2024	Added recommended configuration for Single momentary switch (button) as toggle switch. Added recommended configuration for Single Momentary Switch (Push Button) as White Tone Button. Added recommended configuration for Single Stationary Switch as Automation Switch. Added recommended configuration for Single Momentary Switch (Push Button) as Automation Button. Added examples based upon LD6 accordingly. Other editorial modifications.
3.1	15/01/2024	Modified the short descriptions for Single momentary switch (button) as toggle switch. Modified the short descriptions for Single Momentary Switch (Push Button) as White Tone Button. Modified the short descriptions for Stationary Switch as Automation Switch. Modified the short descriptions for Single Momentary Switch (Push Button) as Automation Button. Other editorial modifications.

5. Contact

ubisys technologies GmbH
Neumannstr. 10
40235 Düsseldorf
Germany

T: +49. 211. 54 21 55 - 19
F: +49. 211. 54 21 55 - 99

www.ubisys.de
info@ubisys.de