GIRA

Erstellung von Templates

1. Was ist ein eine Funktionsvorlage?

Dieses Dokument beschreibt die Erstellung von Templates für den QuadClient.

Im Text wird der Begriff „Template“ für die Funktionsvorlagen verwendet.
In einem Template wird das Aussehen (Darstellung) und Verhalten der markierten Flächen definiert.
Weiterhin werden auch die PopUps damit definiert, die über eine Fläche aufgerufen werden können.
Die Flächen links und in der Mitte dienen der Anzeige und Bedienung.

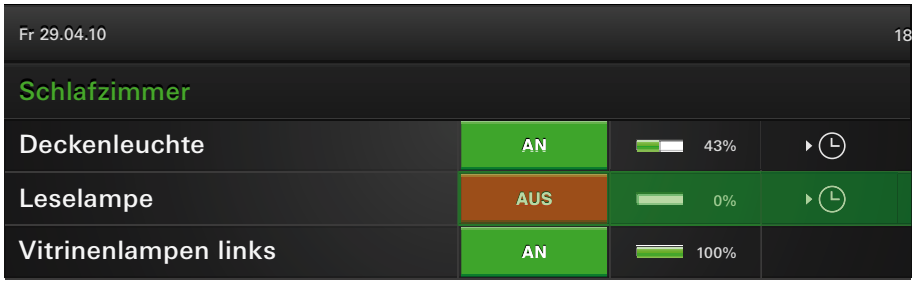


Abb. 1. PopUp

Die rechte Fläche ruft die Zeitschaltuhr auf (optional).
Ein Template enthält die Definitionen für ein Design.
Die Templates werden zur Verarbeitung im QuadConfig importiert. (Programmpunkt System – Gerätevorlagen).
In dieser Dokumentation wird der Aufbau beschrieben.

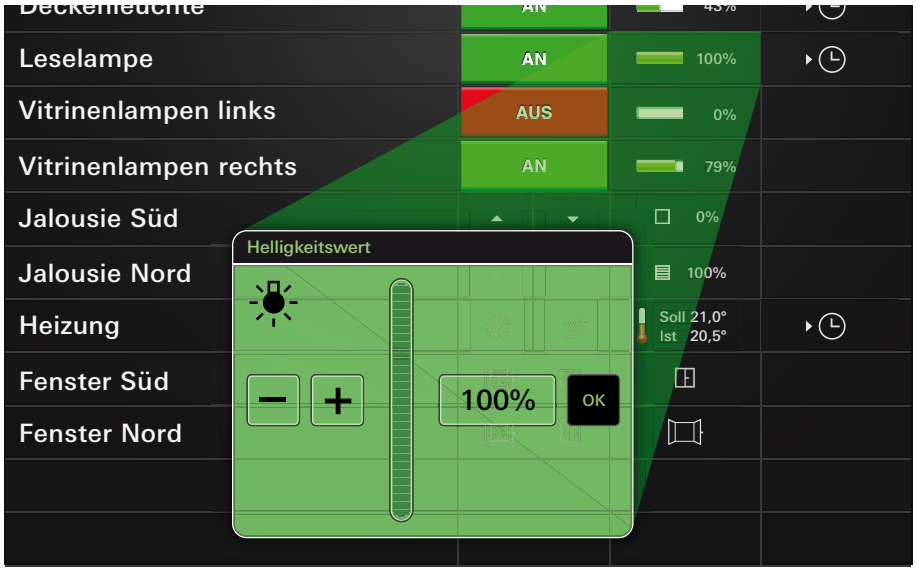


Abb. 2. Popup

PopUp für Dimmer: Jedes PopUp wird zentriert über den linken Bereich gelegt.

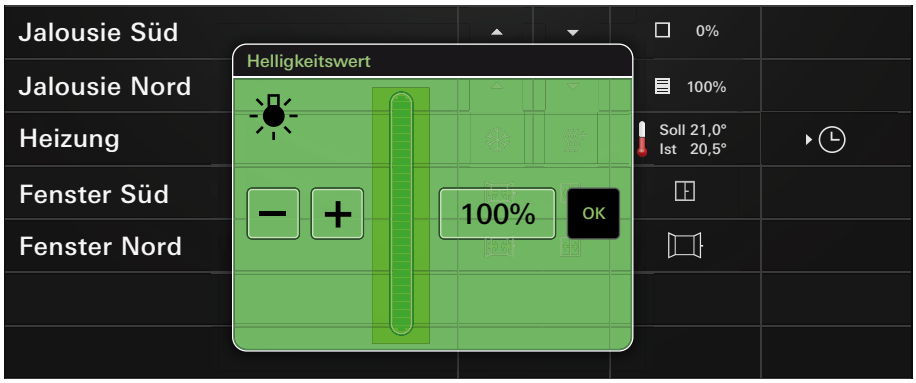


Abb. 3. Popup Slider

Beispiel für PopUp: Sensitive Fläche mit „Füllung“.

Abhängig von der angetippten Position (Abstand von unten) wird die Fläche gefüllt und der Wert (in diesem Beispiel Prozentwert 57%) an ein Eingabefeld übergeben

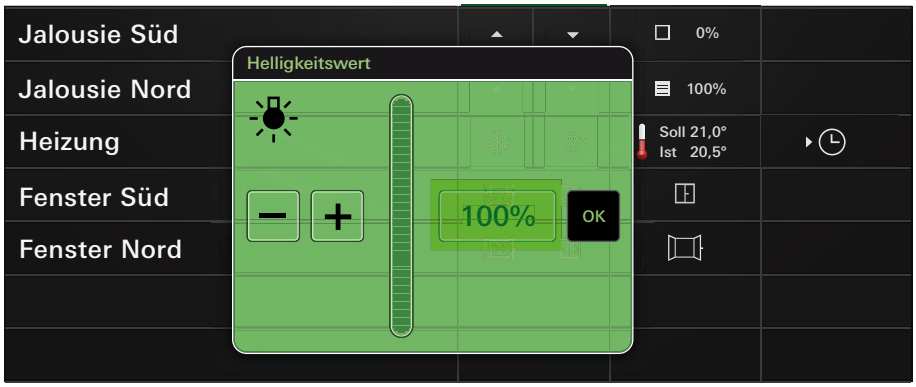


Abb. 4. PopUp numerische Eingabemaske (Edit)

Aufruf der numerischen Eingabemaske durch Antippen der Werteanzeige

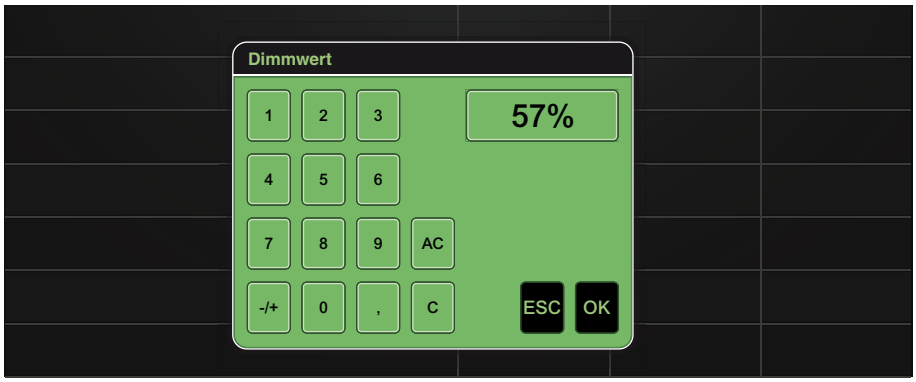


Abb. 5. PopUp Werteingabe

2. Aufbau eines Templates

2.1. Namenskonvention

Name: [Entw.-ID]-[Fortlaufende Nr]_[Beschreibungs-Text].hst
Beispiel: 0-1_Dimmer.hst:
Die Datei-Extension lautet .hst.
Ein Template besteht aus einer zip-Datei.
Diese muss in .hst umbenannt werden.
Der Aufbau wird im Folgenden beschrieben.

2.2. Entwickler ID

0- Diese wird von DaCom Database Computing GmbH verwaltet/vergeben und muss eindeutig sein.
Siehe Verzeichnis der Entwickler IDs

2.3. Fortlaufende Nummer

1-999 Fortlaufende Nummer des Templates. Diese muss über die Gesamtheit der Templates UND Menü-Symbole des Entwicklers in Kombination mit der Entwickler-ID eindeutig sein.

2.4. Fortlaufende Nummer

1-999 Fortlaufende Nummer des Templates. Diese muss über die Gesamtheit der Templates UND Menü-Symbole des Entwicklers in Kombination mit der Entwickler-ID eindeutig sein.

2.5. Bezeichnung

"Dimmer"
Beliebige sprechende Bezeichnung des Templates.
Anmerkung: Wird beim Import im Dateidialog angezeigt.
Anmerkung: 0-1 wird als Template-ID bezeichnet.

2.6. Verzeichnisstruktur

Die Templates (".hst"-Dateien) befinden sich nach dem Import über den Experten-Menüpunkt: "QuadClient/Funktionsvorlagen importieren" in folgendem Verzeichnis-Pfad:

\quad\templates\src

und in diesem Verzeichnis befinden sich die Templates entpackt:

\quad\templates\data

Die oben genannten Verzeichnis-Pfade finden Sie hier:

- Experte v2.3.x: im Verzeichnis, in das der HS/FS Experte installiert wurde.
- Experte v2.4:
- Win7 / Vista: C:\Benutzer\Öffentlich\Öffentliche Dokumente\HS+FS Experte
- WinXP: C:\Dokumente und Einstellungen\All Users\Gemeinsame Dokumente\HS+FS Experte
- Experte v2.5 und höher:
- Win7 / Vista: C:\Benutzer\Öffentlich\Öffentliche Dokumente\HS+FS Experte 2.x
- WinXP: C:\Dokumente und Einstellungen\All Users\Gemeinsame Dokumente\HS+FS Experte 2.x

2.7. Aufbau der *.hst-Datei

- template.xml
- version.xml
- interface.xml
- config_lang.xml
- language.xml
- Ordner thumbs
- Ordner help
- Ordner design_n

3. Template-Definitionsdatei (template.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<template>
  <panel>
    ...
  </panel>
  <popup>
    ...
  </popup>
  <timer>
    ...
  </timer>
</template>
```

Aufbau einer template-Definitionsdatei

3.1. Positionierung von Elementen

Die Positionierung der Elemente eines Templates erfolgt über vordefinierte Layouts. In einem Layout werden verschiedene Zellen verwaltet. Jedes Element, das in einem Template definiert ist muss immer auf ein Layout bzw. die Zelle eines Layouts verweisen.

In einer Layout-Zelle werden neben den Koordinaten für die Positionierung auch Informationen wie Schriftart, Schriftfarbe hinterlegt.

Der Name des Layouts und der Zelle kann entweder vollständig über das Attribut „layout“ mit der Syntax „layout.zelle“ definiert werden oder auch getrennt über die Attribute „layout“ und „cell“

Die beiden folgenden Beispiele sind gleichbedeutend:

```
layout="abc.efg"
layout="abc" cell="efg"
```

3.2. Parameterübergabe aus den Konfigurationsdaten des Projektes

Templates lassen sich über das Konfigurationstool parametrieren. Welche Parameter in einem Template verwendet werden können, wird in der Definitionsdatei „interface.xml“ festgelegt. Ob der in einer Template-Definition zugewiesene Wert aus einem Parameter kommt, wird über den Prefix „$“ gesteuert.

Beispiele:

```
value="$par_skip" <!-- Wert kommt aus einem Parameter-->
value="abc"       <!-- Der Wert ist „abc"--> />
```

3.3. Panel

In der Template-Definitionsdatei werden für das Panel die Flächen 1 und 2 beschrieben, sowie die Bezeichnungen für die in einer Schaltuhr verwendbaren Aktionen festgelegt. Die Bezeichnung des Templates sowie das Aktivieren der Schaltuhr erfolgt über die Konfiguration.

Jede Panel-Fläche setzt sich aus Grafik und (oder) Text zusammen. Textelemente und auch Grafikelemente können dynamisch durch Kommunikationsobjekte veränderbar sein, sie können aber auch durch die Konfiguration des Templates definiert werden.

Zu einer Panel-Fläche können Aktionen festgelegt werden, die beim Click auf die entsprechende Fläche oder Flächenhälfte ausgeführt werden. Aktionen können direkt Kommunikationsobjekte beeinflussen oder PopUps öffnen.

```
<panel pos="n" click="m" layout ="xxx" >
```

3.3.1. Attribut “pos”

Über das Attribut “pos” wird festgelegt auf welche Panelpositon (1 oder 2) sich der folgende Definitionsabschnitt bezieht.

3.3.1.1. Attribut “click”

Legt das Verhalten bei Click auf die Panel-Fläche fest.

- 0 = keine Aktion bei click (nur Darstellung)
- 1 = die ganze Panelfäche ist sensitiv
- 2 = nur die linke Hälfte ist sensitiv
- 3 = nur die rechte Hälfte ist sensitiv
- 4 = die Panelfläche

3.3.2. Attribut “layout” (optional)

Über das Layout werden die Positionen der im Panel definierten Elemente platziert. Ist das Attribut „layout“ bereits mit dem Panel definiert, so müssen die im Panel befindlichen Elemente nicht mehr auf das Layout verweisen und es kann direkt die Layout-Zelle mit dem Attribut „cell“ angegeben werden.

3.4. Die Elemente, aus denen ein Panel zusammengesetzt wird

Im folgenden wird beschreiben, aus welchen Elementen ein Panel zusammengesetzt werden kann.

3.4.1. Element „text“

Mit dem Element “text” kann statischer Text, der über das Attribut „value“ festgelegt wird, dargestellt werden.

```
<text cell="yyy" value="#abc"/>
```

3.4.1.1. Attribut “cell”

Über das Attribut “cell” wird die Zelle des Layouts, auf die das Element referenzieren soll, definiert

Beispiel:

```
cell="efg"
```

3.4.1.2. Attribut “layout” (alternativ)

Soll das mit dem Panel übergebene Layout für dieses Element nicht verwendet werden, oder wurde kein Layout zusammen mit dem Panel definiert, so kann das Layout für jedes Element im Panel überschreiben werden. Die Zelle, die für das Element referenziert werden soll, kann zusammen mit dem Layout übergeben werden (Punktnotation).

Beispiel:

```
layout="abc.efg"
```

3.4.1.3. Attribut “value”

Das Attribut „value“ bestimmt den Text, der über das Text-Element dargestellt werden soll.
Der Wert kann direkt angegeben werden, über die Sprachdatei übersetzt werden, oder über einen Parameter aus den Konfigurationsdaten des Projektes übernommen werden.

Beispiele:

```
value="#1-100_beispieltext"
value="Hallo Welt"
value="$par_text"
```

3.4.1.4. Attribut “font” (optional)

Die im Layout vordefinierte Schriftart kann mit dem Attribut „font“ überschrieben werden.

Beispiel:

```
font=""
```

3.4.1.5. Attribut “align” (optional)

Die im Layout vordefinierte Ausrichtung des Textes kann mit dem Attribut „align“ überschreiben werden.

- 0 = linksbündig
- 1 = zentriert
- 2 = rechtsbündig

Beispiel:

```
align="0"
```

3.4.1.6. Attribut “color” (optional)

Die im Layout vordefinierte Schriftfarbe kann mit dem Attribut „color“ überschreiben werden. Die Farbe wird in hexadezimal-Schreibweise für die Farbkanäle R-G-B angegeben

Beispiel:

```
color="#23FFCE"
```

3.4.2. Element „text_dyn“

Mit dem Element „text_dyn“ wird Text dargestellt, der dynamisch über ein K.-Objekt beeinflusst werden kann.

```
<text_dyn cell="def" len="n" prec="m" unit="abc">
  <content value_slot="yyy" />
  <color case_slot="yyy" />
  <visible slot="yyy" />
</text_dyn>
```

3.4.2.1. Attribut “cell”

Siehe Abschnitt „text“

3.4.2.2. Attribut “layout” (optional)

Siehe Abschnitt „text“

3.4.2.3. Attribut “len”

Das Attribut “len” wird bei der Eingabe von Werten verwendet. Es hat in diesem Kontext keine Auswirkung und wird nur der Vollständigkeit halber hier angegeben („len“, „prec“ und „unit“ bilden immer gemeinsam die Formatierungsanweisung.)

3.4.2.4. Attribut “prec”

Das Attribut “prec” gibt die Anzahl der Nachkommastellen für die Formatierung an.

3.4.2.5. Attribut “unit”

Das Attribut “unit” gibt die Einheit an, die hinter dem Wert nachgestellt wird (Postfix).

3.4.2.6. Attribut “default”

Das Attribut “default” gibt den Wert an, der verwendet werden soll, wenn keine andere Vorgabe gültig ist. (case-Anweisung)

3.4.2.7. Verknüpfung mit einem K.-Objekt

Die Verknüpfung mit einem Kommunikationsobjekt erfolgt über die Slot-Belegung im Abschnitt „content“, „color“ oder „visible“.

3.4.3. Element “content” / Attribut “case_slot”

3.4.3.1. [Beispiel: Den angezeigten Wert (Text) über ein K.-Objekt ändern]

Slot mit dem die case-Fälle geprüft werden.

Beispiel:

```
<text_dyn cell="abc" default="aus">
<content case_slot="dim_val">
        <case from="0" from_type="2" to="0" to_type="2" value="aus" />
        <case from="1" from_type="2" to="100" to_type="2" value="ein" />
      </content>
</text_dyn>
```

3.4.4. Element “case”

Das Element case beschreibt einen Fall

3.4.4.1. Attribut „value“

Wert, den der Text annehmen soll, wenn der Vergleichsfall zutreffend ist

3.4.4.2. Attribut „from“

Über das Attribut “from” wird der Wert festgelegt, ab dem der Vergleichswert zutreffend ist.

3.4.4.3. Attribut „from_type“

- Über das Attribut “from_type” wird die Vergleichsart für „from“ festgelegt
- 1 = größer
- 2 = größer gleich

3.4.4.4. Attribut „to“

Über das Attribut “to” wird der Wert festgelegt, bis zu dem der Vergleichswert zutreffend ist

3.4.4.5. Attribut „to_type“

Über das Attribut “to_type” wird die Vergelichsart für „to“ festgelegt

- 1 = kleiner
- 2 = kleiner gleich

3.4.4.6. Element “content” / Attribut “value _slot”

Das mit dem value_slot verbundene K.-Objekt wird gemäß der Formatierung ausgegeben

Beispiel:

```
<text_dyn cell=”abc” len=”2°” prec=”1” unit=”°”>
  <content value_slot=”soll_temp” />
</text_dyn>
```

3.4.5. Element “color” / Attribut “case_slot”

3.4.5.1. [Beispiel: Die Farbe des Textes über ein K.-Objekt ändern]

Slot mit dem die case-Fälle geprüft werden.

```
<text_dyn cell="text" len="3" prec ="0" unit="$par_unit">
  <color case_slot="dim_val">
    <case from="0" from_type="2" to="0" to_type="2" color="#E6E6E6" />
    <case from="1" from_type="2" to="100" to_type="2" color="#B60000" />
  </color>
</text_dyn>
```

3.4.6. Element “case”

Das Element case beschreibt einen Fall

3.4.6.1. Attribut „color“

Farbe, die der Text annehmen soll, wenn der Vergleichsfall zutreffend ist

3.4.6.2. Attribut „from“

Über das Attribut “from” wird der Wert festgelegt, ab dem der Vergleichswert zutreffend ist.

3.4.6.3. Attribut „from_type“

- Über das Attribut “from_type” wird die Vergleichsart für „from“ festgelegt
- 1 = größer
- 2 = größer gleich

3.4.6.4. Attribut „to“

Über das Attribut “to” wird der Wert festgelegt, bis zu dem der Vergleichswert zutreffend ist

3.4.6.5. Attribut „to_type“

Über das Attribut “to_type” wird die Vergelichsart für „to“ festgelegt

- 1 = kleiner
- 2 = kleiner gleich

3.4.7. Element “visible”

3.4.7.1. [Beispiel: Die Sichtbarkeit des Textes über ein K.-Objekt ändern]

Das Element ist sichtbar, wenn die Bedingung erfüllt ist.

Beispiel:

```
<text_dyn cell="text_on" value="$par_on" def_value="#ON">
  <visible slot="dim_s" from="1" from_type="2" to="100" to_type="2" />
</text_dyn>
```

3.4.7.2. Attribut „slot“

Das mit dem Slot verbundene K.-Objekt wird für den Vergleich herangezogen.

3.4.7.3. Attribut „from“

Über das Attribut “from” wird der Wert festgelegt, ab dem der Vergleichswert zutreffend ist.

3.4.7.4. Attribut „from_type“

- Über das Attribut “from_type” wird die Vergleichsart für „from“ festgelegt
- 1 = größere
- 2 = größer gleich

3.4.7.5. Attribut „to“

Über das Attribut “to” wird der Wert festgelegt, bis zu dem der Vergleichswert zutreffend ist

3.4.7.6. Attribut „to_type“

Über das Attribut “to_type” wird die Vergelichsart für „to“ festgelegt

- 1 = kleiner
- 2 = kleiner gleich

3.4.8. Element „icon“

Mit dem Element “icon” wird ein statisches Icon im Panel dargestellt.

```
<icon cell="yyy" ico="abc" state="n" />
```

3.4.8.1. Attribut “cell”

Über das Attribut “cell” wird die Zelle des Layouts, auf die das Element referenzieren soll, definiert

Beispiel:

```
cell="efg"
```

3.4.8.2. Attribut “layout” (alternativ)

Soll das mit dem Panel übergebene Layout für dieses Element nicht verwendet werden, oder wurde kein Layout zusammen mit dem Panel definiert, so kann das Layout für jedes Element im Panel überschreiben werden. Die Zelle, die für das Element referenziert werden soll, kann zusammen mit dem Layout übergeben werden (Punktnotation).

Beispiel:

```
layout="abc.efg"
```

3.4.8.3. Attribut “ico”

Über das Attribut “ico” wird der Name des Icons übergeben, das dargestellt werden soll. Da die unterschiedlichen Endgeräte mit verschiedenen Grafik-Formaten arbeiten, sollte keine Extension der Grafikdatei angegeben werden (.xaml, .png)

Beispiel:

```
ico="abc.efg"
```

3.4.8.4. Attribut “state”

Das Attribut “state” beschreibt den Zustand des Icons bei Betätigung (click). Folgende Zustände können vorgegeben werden:

- 0 = wird immer dargestellt
- 1 = wird dargestellt wenn das Panel gedrückt wurde
- 2 = wird dargestellt wenn linke Hälfte des Panel gedrückt wurde
- 3 = wird dargestellt wenn rechte Hälfte des Panel gedrückt wurde

3.4.9. Element „icon_dyn“

Mit dem Element „icon_dyn“ wird ein Icon dargestellt, das dynamisch über ein K.-Objekt beeinflusst werden kann.

Beispiele:

```
<icon_dyn cell="bar" ico="tpl_dim_bar">
  <width slot="dim_val" min_value="$par_min" max_value="$par_max" />
</icon_dyn>
```

3.4.9.1. Attribut “cell”

Siehe Element <icon>

3.4.9.2. Attribut “layout” (alternativ)

Siehe Element <icon>

3.4.9.3. Attribut “ico”

Siehe Element <icon>

3.4.9.4. Attribut “state”

Siehe Element <icon>

3.4.10. Element “content” / Attribut “case_slot”

3.4.10.1. [Beispiel: Wechsel des Icons über eine Fallunterscheidung:]

Slot mit dem die case-Fälle geprüft werden.

Beispiel:

```
<icon_dyn layout="pan_togglebtn.full" ico="tpl_btn_green_bg">
  <content case_slot="dim_s">
    <case from="1" from_type="2" to="1" to_type="2" ico="tpl_btn_green_bg" />
    <case from="0" from_type="2" to="0" to_type="2" ico="tpl_btn_red_bg" />
  </content>
</icon_dyn>
```

3.4.11. Element “case”

Das Element case beschreibt einen Fall

3.4.11.1. Attribut „value“

Wert, den der Text annehmen soll, wenn der Vergleichsfall zutreffend ist

3.4.11.2. Attribut „from“

Über das Attribut “from” wird der Wert festgelegt, ab dem der Vergleichswert zutreffend ist.

3.4.11.3. Attribut „from_type“

- Über das Attribut “from_type” wird die Vergleichsart für „from“ festgelegt
- 1 = größer
- 2 = größer gleich

3.4.11.4. Attribut „to“

Über das Attribut “to” wird der Wert festgelegt, bis zu dem der Vergleichswert zutreffend ist

3.4.11.5. Attribut „to_type“

Über das Attribut “to_type” wird die Vergelichsart für „to“ festgelegt

- 1 = kleiner
- 2 = kleiner gleich

3.4.12. Strecken und Stauchen des Icons

Das Strecken und Stauchen eines Icons kann über die folgende Elemente gesteuert werden:

- <width>
- <height>

3.4.13. Element “width”

Über das Element kann die Breite des Icons verändert werden.

Beispiel:

```
<icon_dyn cell="bar" ico="tpl_dim_bar">
  <width slot="dim_val" min_value="$par_min" max_value="$par_max" />
</icon_dyn>
```

3.4.13.1. Attribut “slot”

Das mit dem Slot verbundene K.-Objekt gibt den Wert vor, der für das Strecken bzw. Stauchen des Icons herangezogen wird.

3.4.13.2. Attribut “min_value”

Gibt den Minimal-Wert an, der über den Slot geliefert werden kann. (Wird für die interne Berechnung benötigt)

3.4.13.3. Attribut “max_value”

Gibt den Maximal-Wert an, der über den Slot geliefert werden kann. (Wird für die interne Berechnung benötigt)

3.4.13.4. Attribut “from” (optional)

Gibt den Wertebereich „von“ an. Er ist bereits durch das gewählte Layout vorbelegt und muss nicht angegeben werden.

3.4.13.5. Attribut “to” (optional)

Gibt den Wertebereich „bis“ an. Er ist bereits durch das gewählte Layout vorbelegt und muss nicht angegeben werden.

3.4.14. Element “height”

Über das Element “height” kann die Höhe des Icons verändert werden. Die Beschreibung der Attribute ist identisch mit dem Element „width“.

Beispiel:

```
<icon_dyn cell=”bar” ico=”tpl_dim_bar”>
  <height slot=”temp_val” min_value=”0” max_value=”50” />
</icon_dyn>
```

3.4.15. Verschieben des Icons

Das Verschieben eines Icons kann über die folgende Elemente gesteuert werden:

- <left>
- <right>

3.4.16. Element “left”

Über das Element kann die Position des Icons verändert werden. Es wird der Abstand von links verändert.

Beispiel:

```
<icon_dyn cell="bar" ico="tpl_dim_bar">
  <left slot="val_x" min_value="0" max_value="30" />
</icon_dyn>
```

3.4.16.1. Attribut “slot”

Das mit dem Slot verbundene K.-Objekt gibt den Wert vor, der für das Verschieben des Icons herangezogen wird.

3.4.16.2. Attribut “min_value”

Gibt den Minimal-Wert an, der über den Slot geliefert werden kann. (Wird für die interne Berechnung benötigt)

3.4.16.3. Attribut “max_value”

Gibt den Maximal-Wert an, der über den Slot geliefert werden kann. (Wird für die interne Berechnung benötigt)

3.4.16.4. Attribut “from” (optional)

Gibt den Wertebereich „von“ an. Er ist bereits durch das gewählte Layout vorbelegt und muss nicht angegeben werden.

3.4.16.5. Attribut “to” (optional)

Gibt den Wertebereich „bis“ an. Er ist bereits durch das gewählte Layout vorbelegt und muss nicht angegeben werden.

3.4.17. Element “top”

Über das Element kann die Position des Icons verändert werden. Es wird der Abstand von oben verändert.

Beispiel:

```
<icon_dyn cell="bar" ico="tpl_dim_bar">
  <top slot="val_y" min_value="0" max_value="20" />
</icon_dyn>
```

3.4.18. Beschneiden des Icons

Das Beschneiden eines Icons wird über die folgenden Elemente gesteuert:

- <clip_left>
- <clip_right>
- <clip_top>
- <clip_bottom>

3.4.19. Element “clip_left”

Über das Element „clip_left“ wird das Icon von links beschnitten.

3.4.20. Element “clip_right”

Über das Element „clip_right“ wird das Icon von rechts beschnitten.

3.4.21. Element “clip_top”

Über das Element „clip_top“ wird das Icon von oben beschnitten.

3.4.22. Element “clip_bottom”

Über das Element „clip_bottom“ wird das Icon von unten beschnitten.

Die Attribute zu den einzelnen Elementen entsprechen den Elementen zum Stauchen, Strecken und Verschieben eines Icons. Siehe hier die entsprechende Abschnitte.

3.4.22.1. Aktionen

3.4.23. Element „action_up“ „action_down“

Über die Elemente “action_up” und “action_down” kann festgelegt werden, welche Aktionen ausgeführt werden, wen nein Panel betätigt (click) wird.

Als Aktion kann nicht nur das direkte senden eines Wertes auf ein K.-Objekt festgelegt werden, sondern es lassen sich auch PopUp öffnen, hinter denen weitere Funktionen stecken. Neben dem PopUp für eine numerische Werteingabe, können auch eigene PopUps definiert werden.

3.4.23.1. Attribut „type“

Über das Attribut „type“ wird festgelegt, welcher Aktionstyp ein Betätigen des Panels auslösen soll.

- “set”
  Setzt den Tag auf einen absoluten Wert
- “toggle”
  Toggelt den Tag zwischen 0 und dem Wert
- „step_up“
  Erhöht den Wert des Tags um den angegebenen Wert (bis max. border)
- „step_down“
  Vermindert den Wert des Tags um den angegebenen Wert (bis min. border)
- „seq_start“
  Startet die Sequenz
- „seq_stop“
  Stoppt die Sequenz
- „sc_learn“
  Szene lernen
- „sc_recall“
  Szene abrufen
- „sc_offset_down“
  Erniedrigt alle an der Szene beteiligten K-Objekte
  anhand ihrer Schrittweite.
- „sc_offset_up“
  Erhöht alle an der Szene beteiligten K-Objekte
  anhand ihrer Schrittweite
- „sc_list_down“
  Setzt alle an der Szene beteiligten K-Objekte
  auf den vorhergehenden in den Stammdaten definierten Listenwert.
- „sc_list_up“
  Setzt alle an der Szene beteiligten K-Objekte auf den
  nächsten in den Stammdaten definierten Listenwert
- „copy“
  Setzt den Tag auf den Wert eines anderen Tags. Dieser andere Tag wird über das Attribut src definiert.
- „popup“
  Ruft das PopUp mit der id auf
- „popup_num“
  Ruft das numerische Werteingabe-PopUp auf (Wert wird in Slot gespeichert)

3.4.23.2. Attribut „slot“

Über das Attribut „slot“ wird festgelegt welches K.-Objekt mit der Aktion verbunden ist.

3.4.23.3. Attribut „click“

Das Attribut „click“ legt fest, welcher Panelbereich die Aktion auslösen soll.

- 1=das ganze Panel
- 2= die linke Hälfte des Panel
- 3=die rechte Hälfte des Panel

3.4.23.4. Attribut „from_time“ und „to_time“

Kann nur bei action_up verwendet werden.

Das über „from_time“ und „to_time“ definierte Zeitfenster legt fest, wann die Aktion gültig ist. Mit diesen Attributen kann z.B. ein langer und kurzer Tastendruck unterschieden werden.

3.4.23.5. Attribut „value“

Über das Attribut „value“ wird der Wert festgelegt mit dem die Aktion arbeitet.

3.4.23.6. Attribut „border“

Das Attribut „border“ legt die Grenze für die Aktion „step_up“ bzw. „step_down“ fest.

3.4.23.7. Attribut „id“ (popup)

Über das Attribut „id“ wird bei Aktions-Typ „popup“ die ID des PopUp festgelegt, das beim Auslösen der Aktion geöffnet werden soll

3.4.23.8. Attribut „title“ (popup_num)

Legt den Titel für ein numerisches PopUp fest. (nur in Verbindung mit der Aktions-Typ „popup_num“)

3.4.23.9. Attribut „len“ (popup_num)

Legt die Anzahl der Vorkommastellen fest. (nur in Verbindung mit der Aktions-Typ „popup_num“)

3.4.23.10. Attribut „prec“ (popup_num)

Legt die Anzahl der Nachkommastellen fest. (nur in Verbindung mit der Aktions-Typ „popup_num“)

3.4.23.11. Attribut „unit“ (popup_num)

Legt die Einheit fest, die im numerischen PopUp angezeigt wird. (nur in Verbindung mit der Aktions-Typ „popup_num“)

3.4.23.12. Attribut „min“ (popup_num)

Legt den Minimalwert fest, der über das numerische PopUp eingegeben werden kann. (nur in Verbindung mit der Aktions-Typ „popup_num“)

3.4.23.13. Attribut „max“ (popup_num)

Legt den Maximalwert fest, der über das numerische PopUp eingegeben werden kann. (nur in Verbindung mit der Aktions-Typ „popup_num“)

3.5. PopUp

Neben den vordefinierten Sytem-PopUps wie der numerischen Werteingabe, können auch PopUps über die Template-Definitionsdatei erzeugt werden.

PopUps können neben den von den Panels bekannten Basis-Elementen „text“, text_dyn“, „icon“ und „icon_dyn“ auch aus vordefinierten Funktions-Elementen (Controls) zusammengesetzt werden. Weiterhin stehen Elemente zur Verfügung, die das Verhalten bei Bedienung durch den Benutzer steuern.

Folgende Elemente definieren das Verhalten bei Bedienung:

- click
- slide

Es stehen die folgenden Controls zur Verfügung:

- Button
- Edit-Feld
- Slider

Beispiel einer PopUp-Definition:

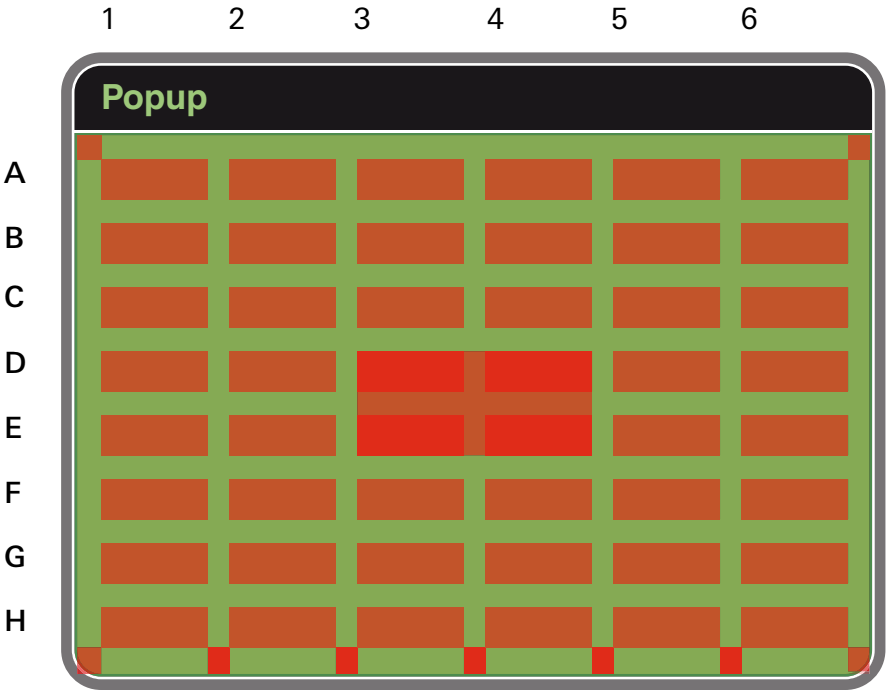
```
<popup id="1" layout="pop_green.form" ico="tpl_pop_bg">
  <!--Kopfzeile Popup-->
  <text layout="pop_green.caption" value="$par_caption" />
  <!--Button |-|-->
  <button layout="grid1.D1-E1+pop_btn_1" type="2" icon="tpl_pop_ico_minus">
    <action_down type="step_down" border="$par_min" slot="dim_val" value="$par_skip" def_value="5" />
  </button>
  <!--Button |+|-->
  <button layout="pop_btn_1" grid="grid1.D2-E2" type="2" icon="tpl_pop_ico_plus">
    <action_down type="step_up" border="$par_max" slot="dim_val" value="$par_skip" def_value="5" />
  </button>
  <!--Slider-->
  <slider layout="slider" grid="grid1.A3-H3" type="1" can_slide="true" slot="dim_val" min="$par_min"
max="$par_max" step="1" />
  <!--Edit-Feld-->
  <edit layout="edit_2" grid="grid1.D4-E5" type="2" title="#1-101_PopupTextDimmer" slot="dim_val" len="3"
prec="0" min="$par_min" max="$par_max" unit="$par_unit" modify="true" />
  <!--OK-Button-->
  <button layout="pop_btn_ok" grid="grid1.D6-E6" type="1" text="#OK">
    <action_up type="close" />
  </button>
</popup>
```

3.5.1. Grids

Die Positionierung der Elemente erfolgt ähnlich einfach wie bei den Panels. Die von den Panels bekannte Positionierung über Layout und Zelle wurde für die PopUps um ein Raster (Grid) erweitert.

Es steht folgendes Grid zur Verfügung:

Grids sind vordefinierte Raster, bei denen jede Zelle eines Rasters die Position und Größe für ein Element mitführt. Elemente können auch über mehrere Zellen definiert werden, wobei jeweils die linke obere Zelle, sowie die rechte untere Zelle angegeben wird.



In der exemplarischen Darstellung eines 6x8 Grids, können maximal 48 Elemente platziert werden. Im Beispiel ist die Zelle D3 mit der Zelle E4 verbunden. Sie kann über „D3-E4“ angesprochen werden.

3.5.2. Zugriff auf eine Zelle des Grids

Der Zugriff auf eine Zelle erfolgt über den Namen des Grids und den Namen der Zelle.

Beispiele:

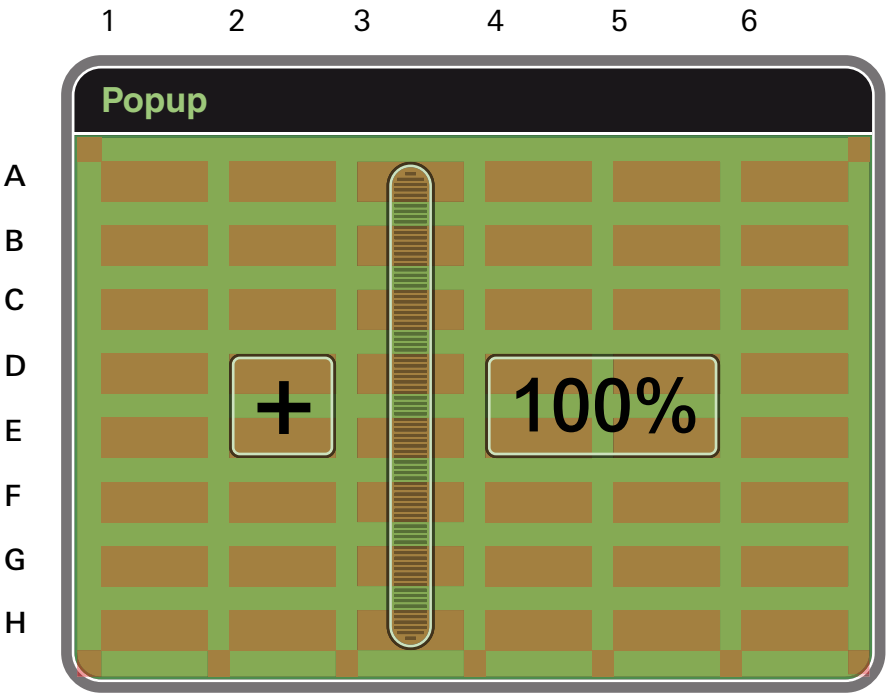
```
grid="grid1.D3"
grid="grid1.D3-E4"
```

3.5.2.1. Kombination aus Grid und Layout

Ein Grid kann lediglich die Position und Dimension einer Zelle zurückgeben. Sollen Texte in einem Grid platziert werden, so müssen über ein Layout die Design-Definitionen wie Textgröße und Textfarbe zugeführt werden.

Entsprechend kann ein Layout auch in einer Grid-Zelle platziert werden. Die tatsächlichen Koordinaten berechnen sich dann aus den Koordinaten der Grid-Zelle und dem Koordinaten der Layout-Zelle.

Beispiel:



Im folgenden Beispiel wird ein Eingabe-Feld (Edit-Control), ein Slider und ein Button platziert. Die Positionen werden über das Grid definiert. Die Elemente werden über die Kombination aus Grid-Zelle und Layout positioniert.

```
<!--Button |+|-->
<button layout="pop_btn_1" grid="grid1.D2-E2" type="2" icon="tpl_pop_ico_plus">
   <action_down type="step_up" border="$par_max" slot="dim_val" value="$par_skip" def_value="5" />
</button>
<!--Slider-->
<slider layout="slider" grid="grid1.A3-H3" type="1" can_slide="false" slot="dim_val" min="$par_min"
max="$par_max" step="1" />
<!--Edit-Feld-->
<edit layout="edit_2" grid="grid1.D4-E5" type="2" title="#1-101_PopupTextDimmer" slot="dim_val" len="3"
prec="0" min="$par_min" max="$par_max" unit="$par_unit" modify="true" />
```

3.5.3. Die Attribute des PopUps

3.5.3.1. Attribut “id”

Das Attribut “id” gibt die eindeutige ID des PopUps an. Die ID wird für die „Action“ des Panel benötigt.

3.5.3.2. Attribut “layout”

Siehe oben

3.5.3.3. Attribut “ico”

Das Attribut “ico” legt den Hintergrund für das PopUp fest.

3.5.4. Elemente für das Verhalten bei Bedienung

3.5.5. Element „click“ in Verbindung mit „action_up“ / „action_down“

Das Element „click“ definiert einen sensitiven Bereich im PopUp

Beispiel:

```
<click grid="grid1.D1-E1" down_ico"icon_pressed">
  <action_down type="set" slot="state" value="2"></action_down>
</click>
```

3.5.6. Element „action_down“ / „action_up“

3.5.6.1. Attribut „type“

Über das Attribut „type“ wird festgelegt, welcher Aktionstyp ein Betätigen des Panels auslösen soll.

- “set”

Setzt den Tag auf einen absoluten Wert

- “toggle”

Toggelt den Tag zwischen 0 und dem Wert

- „step_up“

Erhöht den Wert des Tags um den angegebenen Wert (bis max. border)

- „step_down“

Vermindert den Wert des Tags um den angegebenen Wert (bis min. border)

- „seq_start“

Startet die Sequenz

- „seq_stop“

Stoppt die Sequenz

- „sc_learn“

Szene lernen

- „sc_recall“

Szene abrufen

- „sc_offset_down“

Erniedrigt alle an der Szene beteiligten K-Objekte
anhand ihrer Schrittweite.

- „sc_offset_up“
  Erhöht alle an der Szene beteiligten K-Objekte
  anhand ihrer Schrittweite
- „sc_list_down“
  Setzt alle an der Szene beteiligten K-Objekte
  auf den vorhergehenden in den Stammdaten definierten Listenwert.
- „sc_list_up“
  Setzt alle an der Szene beteiligten K-Objekte auf den
  nächsten in den Stammdaten definierten Listenwert
- „copy“

Setzt den Tag auf den Wert eines anderen Tags. Dieser andere Tag wird über das Attribut src definiert.

- „close“

Schließt das PopUp.

3.5.6.2. Attribut „slot“

Über das Attribut „slot“ wird festgelegt welches K.-Objekt mit der Aktion verbunden ist.

3.5.6.3. Attribut „from_time“ und „to_time“

Kann nur bei action_up verwendet werden.

Das über „from_time“ und „to_time“ definierte Zeitfenster legt fest, wann die Aktion gültig ist. Mit diesen Attributen kann z.B. ein langer und kurzer Tastendruck unterschieden werden.

3.5.7. Element „click“ in Verbindung mit „slide“

Das Element „click“ definiert einen sensitiven Bereich im PopUp

Beispiel:

```
<click grid="grid1.D1-E1">
  <slide slot_vertical="slot_v" value_bottom="0" value_top="100" step_verticel="5"></slide>
</click>
```

3.5.8. Element „slide“

Über das Element „slide“ kann festgelegt werden, welcher Wert auf das verbundene K.-Objekt gesendet werden soll, wenn ein bestimmter Bereich innerhalb der definierten Grid-Zelle berührt wird.

3.5.8.1. Attribut „slot_vertical“

Slot in dem der vertikale Wert gespeichert wird

3.5.8.2. Attribut „slot_horizontal“

Slot in dem der horizontale Wert gespeichert wird

3.5.8.3. Attribut „value_top“

Der Wert den der Slot am oberen Rand annehmen soll

3.5.8.4. Attribut „value_bottom“

Der Wert den der Slot am unteren Rand annehmen soll

3.5.8.5. Attribut „value_left“

Der Wert den der Slot am linken Rand annehmen soll

3.5.8.6. Attribut „value_right“

Der Wert den der Slot am rechten Rand annehmen soll

3.5.8.7. Attribut „step_vertical“

Vertikales Raster für den Slot-Wert

3.5.8.8. Attribut „step_horizontal“

Horizontales Raster für den Slot-Wert

3.5.9. Element „click“ in Verbindung mit „action_num“

Das Element „click“ definiert einen sensitiven Bereich im PopUp

Beispiel:

```
<click grid="grid1.D1-E1">
  <action_num slot="val" title="Soll" len="3" prec="0" unit="%" min="20" max="30"></action_num>
</click>
```

3.5.10. Element „action_num“

Über das Element „action_num“ kann aus einem PopUp das PopUp für die numerische Werteingabe geöffnet werden.

3.5.10.1. Attribut „slot“

Über das Attribut „slot“ wird festgelegt welches K.-Objekt mit der Aktion verbunden ist.

3.5.10.2. Attribut „title“

Legt den Titel für ein numerisches PopUp fest

3.5.10.3. Attribut „len“

Legt die Anzahl der Vorkommastellen fest

3.5.10.4. Attribut „prec“

Legt die Anzahl der Nachkommastellen fest.

3.5.10.5. Attribut „unit“

Legt die Einheit fest, die im numerischen PopUp angezeigt wird.

3.5.11. Attribut „min“

Legt den Minimalwert fest, der über das numerische PopUp eingegeben werden kann

3.5.11.1. Attribut „max“

Legt den Maximalwert fest, der über das numerische PopUp eingegeben werden kann

3.6. Controls

3.6.1. <button>

Ein Button ist ein weiter spezifiziertes <click>-Element und setzt sich aus den folgenden Elementen zusammen:

- Icon für den Hintergrund (Typ icon)
- Icon für den Status „selected“ (Typ icon_dyn)
- Text (optional) für die Beschriftung des Buttons (Typ text)
- Icon (optional) für die Beschriftung des Buttons (Typ icon)
- Icon für den Status „down“ (Typ icon)

Der Button wird über das Grid eines PopUp positioniert. Die Detail-Informationen werden aus den Layout-Daten ermittelt. Im Layout werden die Icons und deren Positionen, aus denen sich ein Button zusammensetzt definiert.
Die auszuführende Aktion ist aus der Definition vom Element <click> zu entnehmen.

In der Template-Definition eines Buttons können die im Layout definierten Icon-Eigenschaften des Buttons überschrieben werden. Wird keine Icon angegeben, so greift der Button auf die im Layout definierten Eigenschaften zu.

Beispiel:

```
<!--Button |-|-->
<button grid="popup.C1" layout="button_1" type="2" icon="tpl_icon_minus" selected_slot="dim_val"
selected_from="1" selected_to="50" >
    <action_down type="step_down" border="0" slot="dim_val" value="$step_width"  def_value="1" />
</button>

<!--Button |+|-->
<button grid="popup.C2" layout="button_1" type="2" icon="tpl_icon_plus">
   <action_down type="step_up" border="100" slot="dim_val" value="$step_width" def_value="1" />
</button>

<!--OK-Button-->
<button grid="popup.C6" layout="button_1" type="1" text="#OK">
   <action_up type="close" />
</button>
```

3.6.2. Die Attribute in einer Button-Definition:

3.6.2.1. <grid>

Hier wird angegeben, an welcher Position im PopUp der Button platziert wird. Die Größe des Buttons richtet sich nach der Grid-Zelle. Sollte im Layout eine Größe über die Attribute „width“ und „height“ angegeben sein, so wird diese Größe für den Button verwendet. . Ist im Layout eine Position über „left“ und „top“ angegeben, so wird die Position um diese Werte in der Grid-Zelle verschoben.

3.6.2.2. <layout>

Hier wird das Layout bestimmt, auf das sich der Button beziehen soll.

3.6.2.3. <icon_bg>

Über dieses Attribut kann die Grafik für den Hintergrund definiert werden. Ist nichts angegeben, so wird die Grafik, die im Layout definiert ist verwendet

3.6.2.4. <type>

Der Typ eines Buttons bestimmt, ob die Beschriftung ein Icon, ein Text oder die Kombination aus Icon und Text ist

- 1 = Text
- 2 = Icon
- 3 = Icon und Text

Achtung: Da bei einem Button-Layout die Positionen für Text und Icon je nach Typ unterschiedlich sein kann, wird bei der Bezeichnung der Layout-Zelle der Typ nachgestellt.

Da bei einem Button-Layout die Positionen für Text und Icon je nach Typ unterschiedlich sein kann, wird bei der Bezeichnung der Layout-Zelle der Typ nachgestellt.

Die Bezeichnung der Layout-Zellen ergibt sich somit wie folgt:

Für einen kleinen Button, der entweder Text oder Icon darstellen kann:

- text1
- icon2

Für einen großen Button, der entweder Text oder Icon oder beides darstellen kann:

- text1
- icon2
- text3
- icon3

3.6.2.5. <icon>

Icon gibt das Icon für die Beschriftung an

3.6.2.6. <text>

Gibt den Text für die Beschriftung an

3.6.2.7. <icon_selected>

Über dieses Attribut kann ein Icon für den Selected-Zustand angegeben werden. Ist nichts angegeben, so wird die Grafik, die im Layout definiert ist verwendet.

3.6.2.8. <selected_slot>

Der selected_slot gibt das K.-Objekt an, das den Selected-Zustand steuert. Der Selected-Zustand ist aktiv, wenn der Wert des K.-Objektes des Wert „selected_value“ entspricht, oder sich in den Grenzen zwischen „selected_from“ und „selected_to“ befindet.

3.6.2.9. <selected_value>

Über dieses Attribut kann der Wert angegeben werden, der den Selected-Zustand aktiviert. (Alternativ kann auch ein Bereich über „selected_from“ und „selected_to“ angegeben werden.)

3.6.2.10. <selected_from> <selected_to>

Über diese Attribute kann ein Bereich aktiviert werden, in dem der Selected-Zustand aktiviert wird.

(Alternativ kann auch ein einzelner Wert über „selected_value“ angegeben werden.)

3.6.3. <slider>

Ein Slider basiert auf dem <click>-Element mit der Funktion „slide“. Der Slider setzt sich aus den folgenden Elementen zusammen:

- Icon für den Hintergrund (Typ icon)
- Icon für den Bargraph (Typ icon_dyn)
- Icon für den Knopf (Typ icon_dyn)

Beispiel:

```
<slider grid="popup.A3-E3" layout="slider_1" type="1" icon_knob="tpl_testknob" slot="dim_val" can_
slide="true" min="0" max="100" step="1"/>
```

3.6.4. Die Attribute in einer Slider-Definition:

3.6.4.1. <grid>

Hier wird angegeben, an welcher Position im PopUp der Slider platziert wird. Die Größe des Sliders richtet sich nach der Grid-Zelle. Sollte im Layout eine Größe über die Attribute „width“ und „height“ angegeben sein, so wird diese Größe für den Slider verwendet. Ist im Layout eine Position über „left“ und „top“ angegeben, so wird die Position um diese Werte in der Grid-Zelle verschoben.

3.6.4.2. <layout>

Hier wird das Layout bestimmt, auf das sich der Slider beziehen soll.

3.6.4.3. <type>

Der Typ eines Sliders bestimmt, wie die Ausrichtung und Arbeitsweise ist.

- 1 = von unten nach oben (z.B. für Dimmer)
- 2 = von oben nach unten (z.B. für Jalousie)
- 3 = von links nach rechts
- 4 =vom rechts nach links

Achtung: Je nach Typ kann der Slider vertikal oder horizontal angeordnet sein. Entsprechend wird der Name der Layout-Zelle gesetzt.

Je nach Typ kann der Slider vertikal oder horizontal angeordnet sein. Entsprechend wird der Name der Layout-Zelle gesetzt.

Für den Typ 1 und 2 wird ein vertikaler Slider angelegt und auf die folgenden Layout-Zellen zugegriffen:

- bg_v
- bar_v
- knob_v

Für den Typ 3 und 4 wird ein vertikaler Slider angelegt und auf die folgenden Layout-Zellen zugegriffen:

- bg_v
- bar_v
- knob_v

3.6.4.4. <can_slide>

Steuert, ob eine Wertverstellung über das Verschieben des Griffels erfolgen kann.

Diese Option steht im QuadClient (Vers. 1.4) noch nicht zur Verfügung.

3.6.4.5. <icon_bg>

Über dieses Attribut kann die Grafik für den Hintergrund definiert werden. Ist nichts angegeben, so wird die Grafik, die im Layout definiert ist verwendet

3.6.4.6. <icon_bar>

Über dieses Attribut kann die Grafik für den Bargraph definiert werden. Ist nichts angegeben, so wird die Grafik, die im Layout definiert ist verwendet

3.6.4.7. <icon_knob>

Über dieses Attribut kann die Grafik für den Griffel des Sliders definiert werden. Ist nichts angegeben, so wird die Grafik, die im Layout definiert ist verwendet

3.6.4.8. <slot>

Gibt den Slot an, der mit der Steuerung des Sliders verbunden werden soll.

3.6.4.9. <min>

Gibt den Minimalwert für den Slot an. Bei diesem Wert ist der Bargraph nicht mehr zu sehen (Auch über Parameter nutzbar)

3.6.4.10. <max>

Gibt den Maximalwert für den Slot an. Bei diesem Wert ist der Bargraph vollständig zu sehen. (Auch über Parameter nutzbar)

3.6.4.11. <step>

Gibt die Stepweite an, mit der sich der Slider positionieren lässt. (Auch über Parameter nutzbar)

3.6.5. <edit>

Das Edit-Control basiert auf dem <click>-Element mit der Funktion „action_num“.
Das Edit-Control setzt sich aus den folgenden Elementen zusammen:

- Icon für den Hintergrund (Typ icon)
- Text für den anzugeigenden Wert (Typ text_dyn)
- Text für eine Beschriftung innerhalb des Edit-Control (Typ text)

Beispiel:

```
<edit grid="popup.C4-C5" layout="edit_1" type="2" title="#0-1_PopupTextDimmer" slot="dim_val" len="3"
prec="0" min="0" max="100" unit="%" info="Soll"/>
```

3.6.6. Die Attribute in einer Edit-Definition:

3.6.6.1. <grid>

Hier wird angegeben, an welcher Position im PopUp das Edit-Control platziert wird. Die Größe des Edit-Controls richtet sich nach der Grid-Zelle. Sollte im Layout eine Größe über die Attribute „width“ und „height“ angegeben sein, so wird diese Größe für das Edit-Control verwendet. Auch eine Verschiebung in der Gridzelle über „left“ und „top“ ist erlaubt.

3.6.6.2. <layout>

Hier wird das Layout bestimmt, auf das sich das Edit-Control beziehen soll.

3.6.6.3. <type>

Der Typ eines Edit-Control bestimmt, wie die Darstellung erfolgen soll

- 1 = es wird nur der Wert dargestellt
- 2 = es wird der Wert und eine Info (oben links) dargestellt

3.6.6.4. <icon_bg>

Über dieses Attribut kann die Grafik für den Hintergrund definiert werden. Ist nichts angegeben, so wird die Grafik, die im Layout definiert ist verwendet

3.6.6.5. <down_ico>

Über dieses Attribut kann die Grafik für den Down-Status definiert werden. Ist nichts angegeben, so wird die Grafik, die im Layout definiert ist verwendet

3.6.6.6. <title>

Über dieses Attribut wird der Titel für das PopUp, sowie der Titel für das numerische Eingabe-PopUp definiert

3.6.6.7. <slot>

Über dieses Attribut wird der Slot definiert, der den darzustellenden Wert repräsentiert.

3.6.6.8. <len>

Über dieses Attribut wird die Länge für den editierbaren Wert definiert

3.6.6.9. <prec>

Über dieses Attribut werden die editierbaren und angezeigten Nachkommastellen festgelegt

3.6.6.10. <min>

Über dieses Attribut wird der Minimalwert definiert, der im Eingabe-PopUp vorgegeben werden kann. (Auch über Parameter nutzbar)

3.6.6.11. <max>

Über dieses Attribut wird der Maximalwert definiert, der im Eingabe-PopUp vorgegeben werden kann. (Auch über Parameter nutzbar)

3.6.6.12. <unit>

Über dieses Attribut wird die angezeigte Einheit für Anzeige und Eingabe-PopUp definiert. (Auch über Parameter nutzbar)

3.6.6.13. <info>

Über dieses Attribut wird der Text vorgegeben, der im Edit-Control erscheinen soll (type=2)

3.6.6.14. <modify>

Über dieses Attribut wird festgelegt, ob der Wert in dem Edit-Feld editiert werden kann (numerisches PopUp bzw. Werteingabe im iPhone/iPad)

3.7. Timer

Die Aktionen, die über eine Timer-Funktion ausgelöst werden können, müssen in der in der Template-Definitionsdatei aufgelistet werden. Die eigentliche Beschreibung der Aktionen , die durch den Timer ausgelöst werden können erfolgt in der Interface-Definitionsdatei.

```
<timer>
  <action text="#ON"/>
  <action text="#OFF"/>
</timer>
```

3.7.1. Element „action“

Listet jeweils eine Aktion auf

3.7.1.1. Attribut „text“

Bezeichnet die Aktion.

4. Interface-Definitionsdatei (interface.xml)

In der „interface.xml“ werden alle Informationen hinterlegt, die für die Konfiguration eines Templates notwendig sind. Es wird festgelegt, welche Kommunikationsobjekte, Szenen und Sequenzen zugeordnet werden können und welche Parameter über Konfigurationstool veränderbar sind. Weiterhin können die Aktionen festgelegt werden, die über eine dem Template zugeordnete Universalzeitschaltuhren ausgelöst werden können.

Aufbau der XML-Datei:

```
<?xml version="1.0" encoding="utf-8"?>
<template>
  <slots>
    <slot/>
    <slot/>
    ...
    <iconset/>
    <iconset/>
    ...
    <parameter/>
    <parameter/>
    ...
  </slots>
  <timer>
    <action/>
    <action/>
    ...
  </timer>
</template>
```

4.1. Attribute des Tag „template“

```
<template has_timer="xxx" function_key="yyy" folder="zzz">
```

4.1.1. Attribut “has_timer”

- true: Funktionvorlage verfügt über eine Zeitschaltuhr
- false: Funktionvorlage verfügt über keine Zeitschaltuhr

4.1.2. Attribut “function_key”

Schlüssel zum Einordnen in Gewerke

Folgende Schlüssel werden aktuell verwendet:

- light
- shutter_jalousie
- scene_sequence
- change_toggle
- firing
- window
- door
- garbage

4.1.3. Attribut “folder”

Schlüssel zum Einordnen der Funktionsvorlagen in den entsprechenden Ordner in QuadConfig/System/Funktionvorlagen

Folgende Schlüssel werden aktuell verwendet:

- light
- shutter_jalousie
- scene_sequence
- change_toggle
- fiiring
- door
- window
- text
- misc

Beispiel:

```
<template has_timer="true" function_key="light" folder="light">
```

4.2. Element „slot“

Slots definieren, welche Kommunikationsobjekte, Szenen und Sequenzen zugeordnet werden können.

Slots

```
<slots>
  <slot id="xxx" text="#abc" help="#abc" type="yyy" required="false" />
  ...

  <parameter id="xxx" text="#abc" help="#abc" type="zzz" min="nnn" max="nnn" default="nnn" />
  ...

  <iconset id="n">
    <icon name="#abc"/>
    ...
  </iconset>
  ...

</slots>
```

4.2.1. Attribut „id“

Über das Attribut „id“ kann ein Slot in der Template-Beschreibung referenziert werden.

4.2.2. Attribut „text“

Das Attribut “text” legt die Bezeichnung des Slots fest, die im Konfigurationstool angezeigt wird.

Texte mit dem vorangestellten „#“ werden über die Sprachdatei „config_lang.xml“ übersetzt.

4.2.3. Attribut „help“

Das Attribut “help” legt den Hilfe-Text fest, der im Konfigurationstool dargestellt werden soll.

Texte mit dem vorangestellten „#“ werden über die Sprachdatei „config_lang.xml“ übersetzt.

4.2.4. Attribut „type“

Das Attribut “type” legt den Typ des Slots fest.

Folgende Typen stehen zur Verfügung (Schreibweise beachten):

- KO_NUM (numerisches K-Obj.)
- KO_ALPHA (alphanumerisches K-Obj.)
- SEQ (Sequenz)
- SCENE (Szene)

4.2.5. Attribut „required“

Ob ein Slot belegt werden muss, oder offen bleiben kann, wird über das Attribut „required“ gesteuert. Ohne Angabe von „required“ ist das Belegen des Slots mit einem K.-Objekt gefordert.

- Slot muss belegt werden: `required=“true“` (oder keine Angabe)
- Slot kann belegt werden: `required=“false“`

4.3. Element „parameter“

Mit Hilfe von Parametern lassen sich Templates über das Konfigurationstool parametrieren.

4.3.1. Attribut „id“

Über das Attribut „id“ kann ein Parameter in der Template-Beschreibung referenziert werden.

4.3.2. Attribut „text“

Das Attribut “text” legt die Bezeichnung des Slots fest, die im Konfigurationstool angezeigt wird.
Texte mit dem vorangestellten „#“ werden über die Sprachdatei „config_lang.xml“ übersetzt.

4.3.3. Attribut „help“

Das Attribut “help” legt den Hilfe-Text fest, der im Konfigurationstool dargestellt werden soll.
Texte mit dem vorangestellten „#“ werden über die Sprachdatei „config_lang.xml“ übersetzt.

4.3.4. Attribut „type“

Folgende Parameter-Typen sind zulässig: (Schreibweise beachten)

- INTEGER
- FLOAT
- TEXT
- ICON

4.3.5. Attribut „min“

Das Attribut „min“ legt fest, welcher Minimalwert für den Parameter zulässig ist.

4.3.6. Attribut „max“

Das Attribut „max“ legt fest, welcher Maximalwert für den Parameter zulässig ist.

4.3.7. Attribut „default“

Das Attribut „deault“ legt den Vorgabewert für den Parameter zulässig fest.

4.3.8. Attribut „visible“

Das Attribut “visible” legt fest, ob der Parameter im Konfigurationstool angezeigt werden soll.

4.3.9. Attribut „icon_set“ (nur in Verbindung mit type ICON)

Das Attribut icon_set legt fest, welche Icons für diesen Parameter ausgewählt werden können.

4.4. Element „icon_set“

```
<iconset id="n">
  <icon name="abc"/>
  <icon name="def"/>
</iconset>
```

Über ein Icon-Sets kann eine Sammlung von Icons zusammengestellt werden, die sich einem Parameter von Typ „ICON“ zuweisen lässt. Es können beliebig viele Icon-Sets erstellt werden. Die Zuordnung erfolgt über die ID des Icon-Sets

4.4.1. Attribut „id“

Über das Attribut „id“ kann ein Icon-Set von einem Parameter referenziert werden.

4.4.2. Tag „icon“

Es können beliebig viele Icons einem Icons-Set hinzugefügt werden. Für jedes Icon wird ein neue Tag „icon“ angelegt

4.4.3. Attribut „name“

Der Name des Icons, das dem Icon-Set zugehörig sein soll. Dem Icon-Namen ist keine Extension (xaml oder png) hinzuzufügen. Im Anhang ist eine Auflistung aller Icons zu finden.

Beispiel:

```
<iconset id="1">
  <icon name="tpl_ico_on"/>
  <icon name="tpl_ico_off"/>
  <icon name="tpl_ico_water_on"/>
  <icon name="tpl_ico_water_off"/>
</iconset>
```

4.5. Timer

Die Aktionen, die über eine Timer-Funktion ausgelöst werden können, müssen in der in der Interface-Definitionsdatei festgelegt werden. Dieses geschieht mit dem Element „timer“. Um die Timerfunktion nutzen zu können muss ebenfalls das Attribut „has_timer“ auf „true“ gesetzt werden. Zusätzlich muss in der template.xml ebenfalls die Liste der Timer-Aktionen aufgeführt werden.

Beispiel:

```
<timer>
  <action text="#ON">
    <cmd op="1" slot_id="dim_s" value="1" />
  </action>
  <action text="#OFF">
    <cmd op="1" slot_id="dim_s" value="0" />
  </action>
</timer>
```

4.6. Element „Action“

Über das Element Action wird eine einzelne Aktion definiert. Es können beliebig viele Aktionen aufgeführt werden.

4.6.3.1. Attribut „text“

Das Attribut „text“ definiert die Bezeichnung für die Aktion. Über den Prefix „#“ kann der Text über die Sprachdatei „config_lang.xml“ übersetzt werden.

4.7. Element “cmd“

Das Element „cmd“ legt den Befehl innerhalb einer Aktion fest.

4.7.3.1. Attribut „op“

Liste der Befehle, hier kann wahlweise der numerische Schlüssel oder Textschlüssel verwendet werden:

- 1 = setValue
- 2 = toggleValue
- 3 = step_up (Achtung, bezieht sich auf die Eintragung in den Stammdaten des K-Objektes-Schrittweite.)
- 4 = step_down (wie step_up)
- 5 = list_up (Achtung, bezieht sich auf die Eintragung in den Stammdaten des K-Objektes-Liste.)
- 6 = list_down (wie list_up)
- 7 = addValue
- 8 = seq_start
- 9 = seq_stop
- 10 = sc_learn
- 11 = sc_recall
- 12 = sc_offset_down
- 13 = sc_offset_up
- 14 = sc_list_down
- 15 = sc_list_up
- 16 = copy

Weitere Erklärungen siehe unten {type} (Ausnahmen: step_up; step_down, list_up, list_down)

4.7.3.2. Attribut „slot_id“

ID des Slots auf den sich der Befehl bezieht.

4.7.3.3. Attribut „value“

Wert auf den sich der Befehl bezieht (optional).

4.8. Erstellung von Templates, bei denen unterschiedliche Verhalten ausgewählt werden können

Zu einem Template lassen sich mehrere Template-Definitionsdateien zuweisen. Sind mehrere Definitionsdateien zugewiesen, so kann eine dieser Definitionen über das Verhalten des Templates in der Konfiguration ausgewählt werden.

Werden unterschiedliche Definitionsdateien zugewiesen, so ist es zwingend erforderlich, das alle Definitionen die gleichen Slots aufweisen.

Die Benennung der Templatedefinitionen (template.xml) erfolgt mit einer fortlaufenden Nummerierung z.B. template_1.xml, template_2.xml, usw.

Beispiel:

```
<subtemplates>
  <subtemplate text="#5-200_1Byte_Prozent" />
  <subtemplate text="#5-200_1Byte_ohne_Vorzeichen"/>
  <subtemplate text="#5-200_1Byte_mit_Vorzeichen" />
  <subtemplate text="#5-200_1Byte_Winkel" />
</subtemplates>
```

4.9. Element “subtemplates”

Das Element “subtemplates” leitet den Bereich ein, in dem alle Subtemplates aufgelistet sind.

4.10. Element “subtemplates”

Das Element “subtemplate” definiert jeweils ein Subtemplate

4.10.3.1. Attribut “text”

Das Attribut “text” legt die Bezeichnung für das Subtemplate fest. Über den Prefix „#“ kann der Text über die Sprachdatei „config_lang.xml“ übersetzt werden.

5. Verwendung von Parametern (interface.xml / template.xml)

5.1. Über Parameter konfigurierbare Templates

Templates lassen sich im Konfigurationstool (QuadConfig) parametrieren. Als Parameter können Werte, Texte und Icons verwendet werden.

- INTEGER
- FLOAT
- TEXT
- ICON

In der Datei „interface.xml“ wird hierzu ein Parameter definiert, auf den in der Beschreibungsdatei des Templates „template.xml“ verwiesen werden kann.

Ein Parameter besteht aus den folgenden Attributen:

- id — Name des Parameter, auf den in der template.xml verwiesen werden kann
- text — Bezeichnung des Parameters (QuadConfig). Sollte mit „#“ übersetzt werden
- help — Hilfe-Text des Parameters (für QuadConfig). Sollte mit „#“ übersetzt werden
- type — Typ des Parameters (INTEGER, FLOAT, TEXT, ICON)
- default — Default-Wert für den Parameter
- min — (optional) Minimalwert für den Wert des Parameters
- max — (optional) Maximalwert für den Wert des Parameters
- visible — (optional) steuert die Sichtbarkeit des Parameters im QuadConfig

5.1.3.1. Beispiel 1: (Text als Parameter)

Im Template „Schalten“ soll der Text für die Bezeichnung des Button durch den Projektant angepasst werden können. Der Default-Wert soll „AN“ und „AUS“ sein. Im Konfigurationstool soll aber auch „AUF“ und „ZU“ vorgegeben werden können.

(config_lang.xml)

```
<text key="#1-102_on" default="on">
  <lang id="de" value="EIN" />
  <lang id="en" value="ON" />
</text>
<text key="#1-102_par_on" default="on">
  <lang id="de" value="Text EIN" />
  <lang id="en" value="Text ON" />
</text>
<text key="#1-102_par_on_hlp" default="Text für Taste bei Status EIN">
  <lang id="de" value="Text für Taste bei Status EIN" />
  <lang id="en" value="Text for button at state ON" />
</text>
<text key="#1-102_off" default="off">
  <lang id="de" value="AUS" />
  <lang id="en" value="OFF" />
</text>
<text key="#1-102_par_off" default="off">
  <lang id="de" value="Text AUS" />
  <lang id="en" value="Text OFF" />
</text>
<text key="#1-102_par_off_hlp" default="Text für Taste bei Status AUS">
  <lang id="de" value="Text für Taste bei Status AUS" />
  <lang id="en" value="Text for button at state OFF" />
</text>
```

(interface.xml)

```
<parameter id="par_on" text="#1-102_par_on" help="#1-102_par_on_hlp" type="TEXT" default="#1-102_on" />
<parameter id="par_off" text="#1-102_par_off" help="#1-102_par_off_hlp" type="TEXT" default="#1-102_off"
/>
```

(template.xml)

```
<text_dyn cell="text_on" value="$par_on" def_value="#ON">
    <visible slot="dim_s" from="1" from_type="2" to="100" to_type="2" />
</text_dyn>

<text_dyn cell="text_off" value="$par_off" def_value="#OFF">
    <visible slot="dim_s" from="0" from_type="2" to="0" to_type="2" />
</text_dyn>
```

5.1.3.2. Beispiel 2: (Wert als Parameter)

Im Template „Dimmer“ soll der Skip-Wert (Schrittweite) für die Taste „+“ und Taste „-„ vorgegeben werden können. Per Default soll der Wert „5“ eingestellt sein.

(interface.xml)

```
<parameter id="par_skip" text="#1-102_skip" help="#1-102_skip_hlp" type="INTEGER" min="1" max="10"
default="5" />
```

(template.xml)

```
<button layout="pop_btn_1" grid="grid1.D1-E1" type="2" icon="tpl_pop_ico_minus">
    <action_down type="step_down" border="0" slot="dim_val" value="$par_skip" def_value="5" />
</button>

<button layout="pop_btn_1" grid="grid1.D2-E2" type="2" icon="tpl_pop_ico_plus">
    <action_down type="step_up" border="100" slot="dim_val" value="$par_skip" def_value="5" />
</button>
```

5.1.3.3. Beispiel 3: (Icon als Parameter)

Das Template „Schalten“ soll universell für verschiedene Aufgaben die mit dem EIS-Typ 1 im Gebäudebus bedient werden können, verwendet werden.

So soll die Statusanzeige für die Steuerung der Gartenbewässerung ein geeignetes Icons für den Zustand „Bewässern“ und „nicht Bewässern“ haben, oder die Steuerung des Garagentores ein entsprechende Symbol für den Zustand „offen“ und „geschlossen“.

Zusätzlich soll der Text auf der Schaltfläche im Template mit „AUF“ und „ZU“ oder „OFFEN“ und „GESCHLOSSEN“ angezeigt werden.

5.2. Vorgaben aus den K.-Objekten übernehmen

Sollen Eigenschaften aus den K.-Objekten übernommen werden, so kann die entsprechende Eigenschaft des K.-Objektes als Default-Wert eines Parameters verwendet werden. Wird der Parameter mit den Eigenschaften des K.-Objektes verbunden, so kann er im Konfigurationstool (QuadConfig) nicht verändert werden.

Der Zuweisungsparameter für den Default-Wert besteht aus dem Namen des Slots, von dem die Eigenschaft des K.-Objektes übernommen werden soll und dem Namen der Eigenschaft.

Es können folgende Zuweisungsparameter verwendet werden:

- $ slotname.init (K.-Objekt: „Init.-Wert“)
- $ slotname.min (K.-Objekt: „Min. Wert“)
- $ slotname.max (K.-Objekt: „Max. Wert“)
- $ slotname.steps (K.-Objekt: „Schrittgröße“)

Sobald im Konfigurationstool ein K.-Objekt zugewiesen wird, wird der Default-Wert aus den Vorgaben des K.-Objektes gesetzt. Wenn der Parameter nicht über die „Visible“-Eigenschaft ausgeblendet wird, kann der Projektant den aus den K.-Objekten übernommenen Wert anpassen.

5.2.3.1. Beispiel 1:

Der Skip-Wert soll aus den Parametern des K.-Objektes, das dem Slot „dim_val“ zugewiesen ist übernommen werden. Die passende Eigenschaft aus dem K.-Objekt ist die Vorgabe „Schrittweite“.

(interface.xml)

```
<slot id="dim_val" text="#1-102_hw" help="#1-102_hw_hlp" type="KO_NUM" />

<parameter id="par_skip" text="#1-102_skip" help="#1-102_skip_hlp" type="INTEGER" min="1" max="10"
default="$dim_val.steps" />
```

5.2.3.2. Beispiel 2:

Der Parameter soll wie aus dem obigen Beispiel mit einem zugewiesenen Default-Wert verarbeitet werden. Es soll aber nicht im Konfigurationstool angezeigt werden.

(interface.xml)

```
<slot id="dim_val" text="#1-102_hw" help="#1-102_hw_hlp" type="KO_NUM" />

<parameter id="par_skip" text="#1-102_skip" help="#1-102_skip_hlp" type="INTEGER" min="1" max="10"
default="$dim_val.steps" visible="false" />
```

5.2.3.3. Beispiel 3:

Beim Dimmer sollen die Min.- und Max.-Werte aus dem K.-Objekt verwendet werden. Sind diese mit 0 und 80 definiert, so arbeitet der Slider im PopUp des Templates und die Werteingabe in diesem Bereich.

Der Slider hat bei „Vollausschlag“ den Wert 80. Die Werteingabe akzeptiert nur Werte im Bereich 0..80.

(interface.xml)

```
<parameter id="par_min" text="#1-102_min" help="#1-102_min_hlp" type="FLOAT" min="0"  max="100"
default="$dim_val.min" />
<parameter id="par_max" text="#1-102_max" help="#1-102_max_hlp" type="FLOAT" min="0" max="100"
default="$dim_val.max" />
```

(template.xml)

```
<!--PANEL 2-->
<icon_dyn cell="bar" ico="tpl_dim_bar">
   <width slot="dim_val" min_value="$par_min" max_value="$par_max" />
</icon_dyn>

<!—-Popup-Button |-|-->
<button layout="pop_btn_1" grid="grid1.D1-E1" type="2" icon="tpl_pop_ico_minus">
<action_down type="step_down" border="$par_min" slot="dim_val" value="$par_skip" def_value="5" />
</button>

<!--Popup-Button |+|-->
<button layout="pop_btn_1" grid="grid1.D2-E2" type="2" icon="tpl_pop_ico_plus">
<action_down type="step_up" border="$par_max" slot="dim_val" value="$par_skip"  def_value="5" />
</button>

<!--Popup-Slider-->
<slider layout="slider" grid="grid1.A3-H3" type="1" can_slide="true" slot="dim_val" min="$par_min"
max="$par_max" step="1" />

<!--Popup-Edit-Feld-->
<edit layout="edit_1" grid="grid1.D4-E5" type="2" title="#1-101_PopupTextDimmer" slot="dim_val" len="3"
prec="0" min="$par_min" max="$par_max" unit="%" />
```

5.3. Attribute, die über einen Parameter gesteuert werden können

5.3.3.1. <text>

- value

5.3.3.2. <text_dyn>

- value
- unit

5.3.3.3. <icon>

- value

5.3.3.4. <icon_dyn>

- min_value (dyn-slot)
- max_value (dyn-slot)
- from (dyn-slot)
- to (dyn-slot)

5.3.3.5. <action_up> / <action_down>

- value
- border
- from_time
- to_time

5.3.3.6. <action_num>

- unit
- title
- prec
- len
- min
- max

5.3.3.7. <slide>

- value_top
- value_bottom
- value_left
- value_right
- step_vertical
- step_horizontal

5.3.3.8. <edit>

- title
- len
- prec
- unit
- min
- max

5.3.3.9. <slider>

- min
- max
- step

5.3.3.10. <button>

- value (action)
- border (action)

6. Versions-Informationen (version.xml)

Beinhaltet Informationen zur Identifikation und Verwaltung des Templates im QuadConfig.

6.1. <version>

Versionsnummer für Fehlerreport, erscheint auch im QC-System und QuadConfig

6.2. <id>

Eindeutige Template-ID über alle Templates hinweg

6.3. <name>

Bezeichnung des Templates
Dieser Name kann auch ein Übersetzungsschlüssel aus der config_lang.xml sein.

6.4. <manufacturer>

Name des Entwicklers, erscheint im QC-System und QuadConfig

6.5. <contact>

Kontakt zum Entwickler, erscheint im QC-System und QuadConfig

Beispiel:

```
<?xml version="1.0" encoding="utf-8"?>
<template>
<header>
   <id>0-1</id>
   <version>1.0.081118</version>
   <name>#0-1_tplname</name>
   <manufacturer>DaCom Database Computing GmbH</manufacturer>
   <contact>http://www.dacom-homeautomation.de</contact>
</header>
</template>
```

7. Sprachdatei (language.xml)

Sprachdatei für den QuadClient. Hier werden die deutschen und fremdsprachigen Namen und Labels Keys zugeordnet, mit denen das QuadClient-Programm intern arbeitet.

Struktur ist identisch zu der von config_lang.xml. Beschreibungen siehe dort.

Unterschied: Im QuadClient (language.xml) können momentan bis zu 20 Sprachen verwendet werden, für den QuadConfig (config_lang.xml) sind es derzeit nur 2.

8. Sprachdatei (config_lang.xml)

Sprachdatei für den QuadConfig. Hier werden die deutschen und englischen Namen und Labels Keys zugeordnet, mit denen das QuadConfig-Programm intern arbeitet.

8.1. <text>

Einleitendes xml-Tag für die Beschreibung eines Objekts.

Ein Objekt kann eine Anzeige-Text, ein Drop-Down-Menü-Eintrag, der Name einer XAML-Grafik oder ähnliches sein, dass in unterschiedlichen Sprachen angezeigt werden soll. Parameter für das <Text>-Tag sind {key} und {default}

Der <Text>-Block wird für jedes einzelne zu übersetzende Objekt wiederholt.

Anmerkung: Die Reihenfolge, in der die einzelnen <text>-Blöcke aufgelistet sind, spielt technisch keine Rolle. Allerdings empfiehlt sich eine alphabetische Sortierung, wenn Sie viele Einträge in der Datei haben, damit Sie selbst schneller einzelne Einträge wieder finden.

8.2. {key}

Format 1:
key="#[ID]-[Nr]_[Bezeichnungs-Text]"
Wird verwendet für:

- Alle Objekte in Templates (".hst")
- Texte in Menü-Symbol-Dateien (".hsm") (z.B. Im Import-Programm angezeigter Name der ".hsm"-Datei)

Format 2:
key="#prev.menu.[ID]-[Nr]_[Bezeichnungs-Text]"
Wird verwendet für:

- Grafik- (".xaml") und Preview- (".png") Dateien für Menü-Symbol-Dateien (".hsm")

"ID": Hersteller-ID
"Nr": Laufende Nr. über alle Templates und Menü-Symbol-Dateien des Herstellers.
"Bezeichnungs-Text": Eindeutige Bezeichnung für das Objekt (siehe dazu auch Kapitel "Aufbau des Templates")

8.3. {default}

Angezeigter Text für das Objekt in der Standard-Sprache (normalerweise deutsch). Der {default}-Text wird immer dann angezeigt, wenn eine Sprache zur Anzeige gewählt wurde, die nicht im folgenden <lang>-Tag definiert wurde. Der {default}-Text sollte unbedingt auch im <lang>-Tag in der für den {default}-Tag verwendeten Sprache angelegt werden.

8.4. <lang>

Einleitendes HTML-Tag für die Texte in den verschiedenen Sprachen. Parameter sind {id} und {value}.

8.5. {id}

Format: id="2-stelliges Sprachkürzel".
Momentan werden vom QuadClient folgende Sprachkürzel unterstützt, die Sie in der language.xml verwenden können:

de - "Deutsch"	lv - "Lettisch"	gr - "Griechisch"	en - "Englisch"
nl - "Holländisch"	se - "Schwedisch"	cz - "Tschechisch"	no - "Norwegisch"
hu - "Ungarisch"	dk - "Dänisch"	pl - "Polnisch"	si - "Slovenisch"
es - "Spanisch"	pt - "Portugiesisch"	is - "Isländisch"	fi - "Finnisch"
ro - "Rumänisch"	sk - "Slovakisch"	fr - "Französisch"	ru - "Russisch"

Achtung: Vom QuadConfig werden derzeit nur die Sprachen "de" und "en" unterstützt. Verwenden Sie also in der config_lang.xml nur diese beiden Sprachen!

8.6. {value}

Der anzuzeigende Text in der bei {id} angegebenen Sprache

Beispiel:

```
<?xml version='1.0' encoding='utf-8'?>
<language>
   <text key="#0-1_d" default="Dimmen">
      <lang id="de" value="Dimmen" />
      <lang id="en" value="Dim" />
   </text>
   <text key="#0-1_d_hlp" default="Dimmobjekt des Dimmers">
      <lang id="de" value="Dimmobjekt des Dimmers" />
      <lang id="en" value="Dimming object of dimmer" />
   </text>
   <text key="#0-1_hw" default="Helligkeitswert">
      <lang id="de" value="Helligkeitswert" />
      <lang id="en" value="Brightness value" />
   </text>
   <text key="#0-1_hw_hlp" default="Helligkeitswert des Dimmers">
      <lang id="de" value="Helligkeitswert des Dimmers" />
      <lang id="en" value="Brightness value of dimmer" />
   </text>
    .
    .
</language>
```

9. Layoutbeschreibung

Zur Darstellung von Unterschiedlichen Funktionen sind im QuadClient verschiedene Layouts vordefiniert. Diese können aus der template.xml aufgerufen, und mit Informationen gefüllt werden.

Allgemeine Informationen:

Innerhalb der Layouts sind in manchen Fällen Default-Grafiken hinterlegt. Diese müssen beim Aufruf eines Layouts aus dem Template nicht angegeben werden, können aber ersetzt werden.

Wenn es sich um Defaultzellen handelt ist dies in der Layoutbeschreibung vermerkt. Dies betrifft in erster Linie Gui-Elemente wie Down-Status oder Hintergrundgrafik.

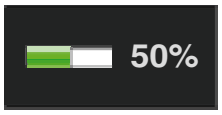
Achtung: Alle Zellen die angezeigt werden sollen müssen aufgerufen werden.
Ausnahme: Down-Zellen
Diese werden je nach Click-Tag automatisch aufgerufen

Die Font-Tags sind in den Layouts definiert und müssen nicht aufgerufen werden.

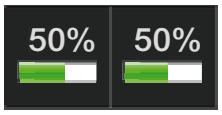
9.1. Übersicht der Layouts

Nachfolgend alle Dateien die zur Nutzung in Templates mit dem Stammdatensatz des QuadClients mitgeliefert werden.

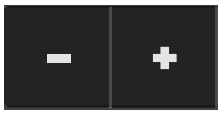
Togglebutton
(pan_togglebutton)



Dimmerbar
(pan_dimbar)

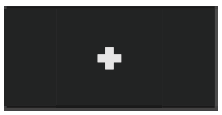


Geteilte Dimmerbar
(pan_doublebar)

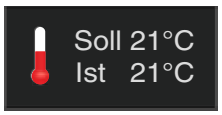


Doppelbutton
(pan_doublebtn)

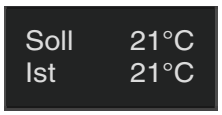
Doppeltogglebutton
(pan_doubletoggle)



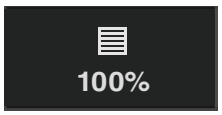
Einzelbutton
(pan_singlebtn)



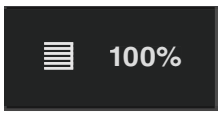
Heizung
(pan_heat)



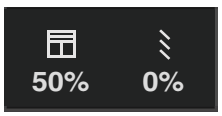
Vier Texte
(pan_4txt)



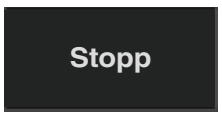
Icon mit Wert vertikal
(pan_ico_val_v)



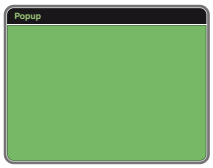
Icon mit Wert horizontal
(pan_ico_val_h)



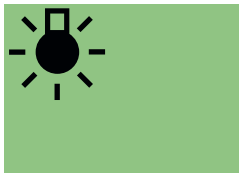
Zwei Icons mit Wert vertikal
(pan_2ico_2val_v)



Textanzeige
(pan_text)



Standard Popup
(pop_green)



Icon in PopUp
(pop_icon)

Slider
(slider)

Editfeld 2 Spalten
(edit_2)

Editfeld 3 Spalten
(edit_3)

PopUpbutton 1 Spalte
(pop_btn_1)

PopUpbutton 2 Spalten
(pop_btn_2)

PopUpbutton 3 Spalten
(pop_btn_3)

PopUpbutton volle Breite
(pop_btn_full)

PopUpbutton OK/ESC
(pop_btn_ok)

9.2. Panellayouts

9.2.1. Togglebutton

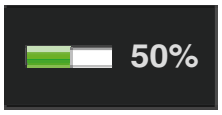
Name: "pan_togglebtn"
Zellen:

- "full"
  Vollflächiger Hintergrund
- "text_on"
  Text für eingeschalteten Zustand
- "text_off"
  Text für ausgeschalteten Zustand
- "down1" (default)
  Overlaygrafik für Downstatus

Einbindungsbeispiel:

```
<panel pos="1" click="1" layout="pan_togglebtn">
    <!--Fläche rot/grün-->
    <icon_dyn layout="pan_togglebtn.full" ico="tpl_btn_green_bg">
      <content case_slot="dim_s">
        <case from="1" from_type="2" to="1" to_type="2" ico="tpl_btn_red_bg" />
        <case from="0" from_type="2" to="0" to_type="2" ico="tpl_btn_green_bg" />
      </content>
    </icon_dyn>
    <!--Text ON / OFF-->
    <text_dyn cell="text_on" value="text_on" def_value="#ON">
      <visible slot="dim_s" from="1" from_type="2" to="100" to_type="2" />
    </text_dyn>
    <text_dyn cell="text_off" value="text_off" def_value="#OFF">
      <visible slot="dim_s" from="0" from_type="2" to="0" to_type="2" />
    </text_dyn>
    <action_down click="1" type="toggle" slot="dim_s" value="1" />
  </panel>
```

9.2.2. Dimmerbar



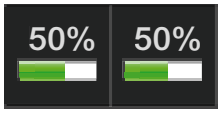
Name: "pan_dimmbar"
Zellen:

- "bg"
  Hintergrund
- "full"
  Vollflächiger Hintergrund (optional)
- "bar"
  Bargraph
- "text"
  Textfeld für Wert
- "down1" (default)
  Overlaygrafik für Downstatus

Einbindungsbeispiel:

```
<panel pos="2" click="1" layout="pan_dimbar">
   <icon cell="bg" ico="tpl_dim_bg" />
   <icon_dyn cell="bar" ico="tpl_dim_bar">
      <width slot="dim_val" min_value="0" max_value="100" />
   </icon_dyn>
   <text_dyn cell="text" value="{0:#0}%">
      <content value_slot="dim_val" />
   </text_dyn>
   <action_down click="1" type="popup" id="1" />
</panel>
```

9.2.3. Geteilte Dimmerbar



Name: "pan_doublebar"
Zellen:

- "left_bg"
  Hintergrund linke Hälfte
- "left_bar"
  Bargraph linke Hälfte
- "left_text"
  Textfeld für Wert linke Hälfte
- "line"
  Trennlinie (default)
- "right_bg"
  Hintergrund rechte Hälfte
- "right_bar"
  Bargraph rechte Hälfte
- "right_text"
  Textfeld für Wert rechte Hälfte
- "down1" (default)
  "down2" (default)
  "down3" (default)
  Overlaygrafik für Downstatus je nach gesetztem Clicktag

9.2.4. Geteilter Button

Name: "pan_doublebtn"
Zellen:

- "leftico"
  Icon linke Seite
- "txt1"
  Textfeld für linken Button (altenativ)
- "rightico"
  Icon rechte Seite
- "txt2"
  Textfeld für rechten Button (altenativ)
- "down1" (default)
  "down2" (default)
  "down3" (default)
  Overlaygrafik für Downstatus je nach gesetztem Clicktag

Einbindungsbeispiel:

```
<panel pos="1" click="4" layout="pan_doublebtn">
   <icon cell="leftico" ico="tpl_ico_min"/>
   <icon cell="line"/>
   <icon cell="rightico" ico="tpl_ico_plus"/>
   <action_up click="2" type="set" slot="jal_kz" value="0" to_time="500" />
   <action_down click="2" type="set" slot="jal_lz" value="0" />
   <action_up click="3" type="set" slot="jal_kz" value="1" to_time="500" />
   <action_down click="3" type="set" slot="jal_lz" value="1" />
</panel>
```

Geteilter Togglebutton

Name: "pan_doubletoggle"
Zellen:

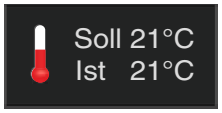
- "left_bg"
  Hintergrund linke Hälfte
- "left_text_on"
  Textfeld linker Button für eingeschalteten Zustand
- "left_text_off"
  Textfeld linker Button für ausgeschalteten Zustand
- "line"
  Trennlinie (default)
- "right_bg"
  Hintergrund rechte Hälfte
- "right_text_on"
  Textfeld rechter Button für eingeschalteten Zustand
- "right_text_off"
  Textfeld rechter Button für ausgeschalteten Zustand
- "down1" (default)
  "down2" (default)
  "down3" (default)
  Overlaygrafik für Downstatus je nach gesetztem Clicktag

9.2.5. Einzelbutton

Name: "pan_singlebtn"
Zellen:

- "ico_w"
  Zelle für breites Icon
- "ico_w"
  Zelle für schmales Icon
- "down1" (default)
  Overlaygrafik für Downstatus

9.2.6. Heizung



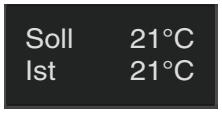
Name: "pan_heat"
Zellen:

- "icon"
  Zelle für Icon
- "txt1"
  Text oben links
- "txt2"
  Text oben rechts
- "txt3"
  Text unten links
- "txt4"
  Text unten rechts
- "down1" (default)
  Overlaygrafik für Downstatus

Einbindungsbeispiel:

```
<panel pos="2" click="1" layout="pan_heat">
   <icon cell="icon" ico="tpl_heat_valico1" />
   <text cell="txt1" value="#1-13_Target" color="#E6E6E6" />
   <text_dyn cell="txt2" value="{0:#0.0}°">
      <content value_slot="soll_temp" />
   </text_dyn>
   <text cell="txt3" value="#1-13_Actual" />
   <text_dyn cell="txt4" value="{0:#0.0}°">
      <content value_slot="ist_temp" />
   </text_dyn>
   <action_down click="1" type="popup" id="1031" />
</panel>
```

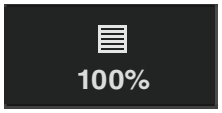
9.2.7. Vier Texte



Name: "pan_4txt"
Zellen:

- "txt1"
  Text oben links
- "txt2"
  Text oben rechts
- "txt3"
  Text unten links
- "txt4"
  Text unten rechts
- "down1" (default)
  Overlaygrafik für Downstatus

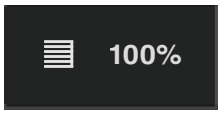
9.2.8. Icon + Wert vertikal



Name: "pan_ico_val_v"
Zellen:

- "txt1"
  Zelle für Wert
- "icon"
  Zelle für Icon
- "down1" (default)
  Overlaygrafik für Downstatus

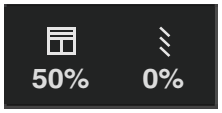
9.2.9. Icon + Wert horizontal



Name: "pan_ico_val_h"
Zellen:

- "txt1"
  Zelle für Wert
- "icon"
  Zelle für Icon
- "down1" (default)
  Overlaygrafik für Downstatus

9.2.10. Zwei Icons + Zwei Werte vertikal



Name: "pan_2ico_2val_v"
Zellen:

- "txt1"
  Zelle für Wert links
- "icon1"
  Zelle für Icon links
- "txt2"
  Zelle für Wert rechts
- "icon2"
  Zelle für Icon rechts
- "down1" (default)
  Overlaygrafik für Downstatus

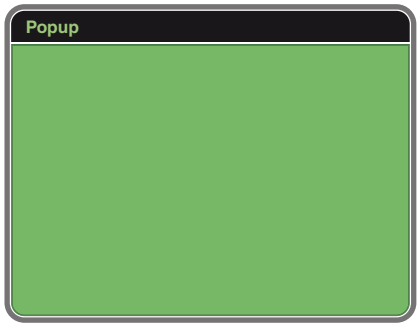
9.2.11. Text

Name: "pan_text"
Zellen:

- "txt"
  Zelle für Text zentriert
- "txt2"
  Zelle für Text oben
- "txt3"
  Zelle für Text unten
- "down1" (default)
  Overlaygrafik für Downstatus

9.3. PopUp Layouts

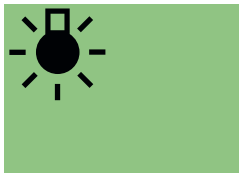
9.3.1. PopUp



Name: "pan_ico_val_h"
Zellen:

- "form"
  Zelle für Hintergrund
- "caption"
  Zelle Titel

9.3.2. PopUp Icon



Name: "pop_icon"
Ausdehnung im Grid: 2x3 Gridzellen
Zellen:

- "icon"
  Zelle für Icon

9.3.3. Slider

Name: "slider"
Feste Komponente (siehe Slider)
Ausdehnung im Grid: 1x8 Gridzellen

Zellen:

- bg - Slider Hintergrund (default)
- bar - Sliderinhalt (default)
- knob - Slidergreifer (default)
- bg_h - Slider Hintergrund horizontal (default)
- bar_h - Sliderinhalt horizontal (default)
- knob_h - Slidergreifer horizontal (default)

Die entsprechenden Zellen werden automatisch je nach Type eingefügt.
Die Variable can_slide legt fest ob im QC der Slidergreifer eingeblendet werden soll.

9.3.4. Edit zwei Spalten

Name: "edit2"
Feste Komponente (siehe Editfeld)
Ausdehnung im Grid: 2x2 Gridzellen

Zellen:

- bg - Hintergrundgrafik (default)
- value - Textfeld groß für Wert
- info - Textfeld klein für Label (optional)
- down - down overlay (default)

9.3.5. Edit drei Spalten

Name: "edit3"
Feste Komponente (siehe Editfeld)
Ausdehnung im Grid: 3x2 Gridzellen

Zellen:

- bg - Hintergrundgrafik (default)
- value - Textfeld groß für Wert
- info - Textfeld klein für Label (optional)
- down - down overlay (default)

9.3.6. Button eine Spalte

Name: "pop_btn_1"
Feste Komponente (siehe Button)
Ausdehnung im Grid: 1x2 Gridzellen

Zellen (werden je nach Typ ausgewählt):

- bg - Hintergrundgrafik (default)
- selected - Zelle für ausgewähltn Zustand (default)
- text1 - Textzelle für Anzeige nur Text
- icon2 - Iconzelle für Anzeige nur Icon
- down - down overlay (default)

9.3.7. Button zwei Spalten

Name: "pop_btn_2"
Feste Komponente (siehe Button)
Ausdehnung im Grid: 2x2 Gridzellen

Zellen (werden je nach Typ ausgewählt):

- bg - Hintergrundgrafik (default)
- selected - Zelle für ausgewähltn Zustand (default)
- text1 - Textzelle für Anzeige nur Text
- icon3 - Iconzelle für Anzeige Text und Icon
- text3 - Textzelle für Anzeige Text und Icon
- icon2 - Iconzelle für Anzeige nur Icon
- down - down overlay (default)

9.3.8. Button drei Spalten

Name: "pop_btn_2"
Feste Komponente (siehe Button)
Ausdehnung im Grid: 2x2 Gridzellen

Zellen (werden je nach Typ ausgewählt):

- bg - Hintergrundgrafik (default)
- selected - Zelle für ausgewähltn Zustand (default)
- text1 - Textzelle für Anzeige nur Text
- icon3 - Iconzelle für Anzeige Text und Icon
- text3 - Textzelle für Anzeige Text und Icon
- icon2 - Iconzelle für Anzeige nur Icon
- down - down overlay (default)

9.3.9. Button volle Breite

Name: "pop_btn_full"
Feste Komponente (siehe Button)
Ausdehnung im Grid: 7x2 Gridzellen

Zellen (werden je nach Typ ausgewählt):

- bg - Hintergrundgrafik (default)
- selected - Zelle für ausgewähltn Zustand (default)
- text1 - Textzelle für Anzeige nur Text
- icon3 - Iconzelle für Anzeige Text und Icon
- text3 - Textzelle für Anzeige Text und Icon
- icon2 - Iconzelle für Anzeige nur Icon
- down - down overlay (default)

9.3.10. Button OK/ESC

Name: "pop_btn_ok"
Feste Komponente (siehe Button)
Ausdehnung im Grid: 1x2 Gridzellen

Zellen (werden je nach Typ ausgewählt):

- bg - Hintergrundgrafik (default)
- text1 - Textzelle für Anzeige nur Text
- down - down overlay (default)

9.4. Liste aller mitgelieferten Grafiken.

Nachfolgend alle Dateien die zur Nutzung in Templates mit dem Stammdatensatz des QuadClients mitgeliefert werden.

tpl_btn_blue_bg
(horizontal skalierbar)

tpl_btn_green_bg
(horizontal skalierbar)

tpl_btn_blue_bg
(horizontal skalierbar)

tpl_btn_blue_bg
(horizontal skalierbar)

tpl_btn_toggle_ol
(default)

tpl_dim_bar

tpl_dim_bar_blue

tpl_dim_bar_red

tpl_dim_bar_yellow

tpl_dim_bg

tpl_doubledim_bg

tpl_heat_antifreeze

tpl_heat_comfort

tpl_heat_cooling

tpl_heat_heating

tpl_heat_heating_cooling

tpl_heat_night

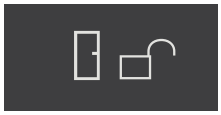
tpl_heat_party

tpl_heat_standby

tpl_heat_valico1

tpl_heat_valico2

tpl_ico_door_locked



tpl_ico_door_unlocked

tpl_ico_down

tpl_ico_jal_0

tpl_ico_jal_1

tpl_ico_jal_2

tpl_ico_jal_3

tpl_ico_jal_4

tpl_ico_jal_lam_0

tpl_ico_jal_lam_1

tpl_ico_jal_lam_2

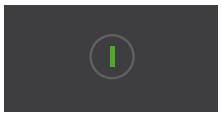
tpl_ico_jal_lam_3

tpl_ico_jal_lam_4

tpl_ico_lock_action


tpl_ico_min


tpl_ico_off


tpl_ico_on


tpl_ico_plus


tpl_ico_sce_play

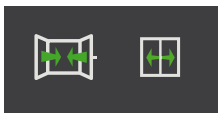

tpl_ico_sce_sav


tpl_ico_up


tpl_ico_water_off


tpl_ico_water_on


tpl_ico_win_closed


tpl_ico_win_motor


tpl_ico_win_open


tpl_ico_win_tilt


tpl_panel_ol
(default)


tpl_pop_bg
(default)


tpl_pop_btn_1
(default)


tpl_pop_btn_1_down_ol
(default)


tpl_pop_btn_1_select
(default)


tpl_pop_btn_2
(default)


tpl_pop_btn_2_down_ol
(default)


tpl_pop_btn_2_select
(default)


tpl_pop_btn_3
(default)


tpl_pop_btn_3_down_ol
(default)


tpl_pop_btn_3_select
(default)


tpl_pop_btn_full
(default)


tpl_pop_btn_full_down_ol
(default)


tpl_pop_btn_full_select
(default)


tpl_pop_btn_ok
(default)


tpl_pop_btn_ok_down_ol
(default)

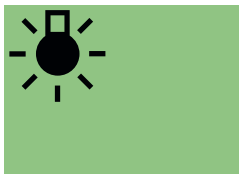

tpl_pop_heat_antifreeze

tpl_pop_heat_comfort

tpl_pop_heat_night

tpl_pop_heat_party

tpl_pop_heat_standby



tpl_pop_ico_dimmer

tpl_pop_ico_down

tpl_pop_ico_heat

tpl_pop_ico_jalousie

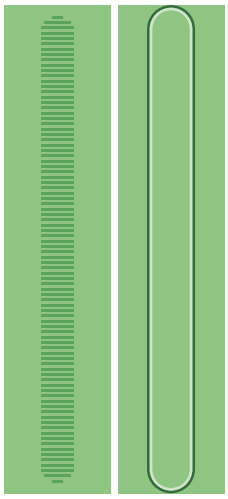
tpl_pop_ico_lamelle

tpl_pop_ico_minus

tpl_pop_ico_plus

tpl_pop_ico_sce

tpl_pop_ico_up



tpl_pop_slider
tpl_pop_slider_bg
(default)

tpl_pop_slider_h
(default)

tpl_pop_slider_h_bg
(default)

tpl_pop_slider_h_knob
(default)

tpl_pop_slider_knob
(default)

11. Anhang 2 - Menüsymbole

Ab Version 2.5 besteht die Möglichkeiten Symbole für die Menüdarstellung im QuadClient zu importieren. Die Dateien haben die Extension ".hsm".
Der Menüpunkt für den Import befindet sich im Experten unter "QuadClient/Symbole importieren".
Die Datei ist eine Zip-Datei und muss in ".hsm" umbenannt werden.
Der Dateiname sollte mit der Entwickler-ID (siehe oben) beginnen.
Anmerkung: Grundsätzlich kann "jederName.hsm" als Dateiname gewählt werden.

Achtung: Wichtig für die Dateinamen der Grafiken: Alle Dateinamen der XAML-Dateien und Vorschaubilder müssen mit der Entwickler-ID (siehe oben) beginnen. Diese Eindeutigkeit bezieht sich auf die Gesamtmenge der Templates und der Menüsymbole!

Beispiel: Sie haben eigene Templates von ID 23-0 bis 23-23 erstellt. Nun wollen Sie ein oder mehrere Menüsymbole erzeugen. Hierzu vergeben Sie dann die nächste ID 23-24.

ALLE in einer ".hsm"-Datei enthaltenen Grafik-Dateien (XAML's und Previews) haben dieselbe ID (hier im Beispiel wäre das dann 23-24)!

Achtung: Eine Preview-Datei muss immer genauso heißen, wie ihre korrespondierende XAML-Datei!

11.1. hsm-Datei (Aufbau/Inhalt)

- config_lang.xml
  Enthält die Klartextbezeichnungen der Symbole
- version.xml
  Enthält Hersteller-Informationen zur Datei (siehe oben: version.xml)
- /preview_de
  Ordner für deutschsprachige Vorschaubilder für den QuadConfig.
- /preview_en
  Ordner für englischsprachige Vorschaubilder für den QuadConfig
- /preview_en
  Ordner für XAML-Dateien, die nur im angegebenen Design verwendet werden sollen (n=ID des Designs, fortlaufende Nr.)
- /xaml
  Ordner für XAML-Dateien, die in allen Designs verwendet werden sollen.

Die Menü-Symbole (".hsm"-Dateien) befinden sich nach dem Import über den Experten-Menüpunkt: "QuadClient/Symbole importieren" in folgendem Verzeichnis-Pfad:

\quad\client\src

und in dieses Verzeichnis (und einige Unterverzeichnisse) werden die Dateien entpackt und teilweise in dort vorhandene ZIP-Archiv-Dateien eingebunden:

\quad\client\data

(Siehe Verzeichisstruktur)

11.2. config_lang.xml für Menüsymbole (weitere Infos: siehe config_lang)

Anmerkung: Damit der Anzeige-Name sprachabhängig angezeigt wird, tragen Sie in der Datei version.xml im Tag "<name>" den Schlüssel aus der 1. "<text>"-Zeile (#0-200_hsm_name) ein.

Anmerkung: Für Menü-Symbole gibt es (im Unterschied zu Templates) KEINE language.xml-Datei!

Beispiel:

```
<?xml version='1.0' encoding='utf-8'?>
<language>
   <text key="#0-200_hsm_name" default="Menü-Symbol Demo-Datei">
      <lang id="de" value="Menü-Symbol Demo-Datei" />
      <lang id="en" value="Sample menu icon file" />
   </text>
   <text key="#prev.menu.0-200_mnu_ico_nr_1" default="Icon Nr 1">
      <lang id="de" value="Icon Nr 1" />
      <lang id="en" value="Icon no 1" />
   </text>
   <text key="#prev.menu.0-200_mnu_ico_nr_2" default="Icon Nr 2">
      <lang id="de" value="Icon Nr 2" />
      <lang id="en" value="Icon no 2" />
   </text>
   <text key="#prev.menu.0-200_mnu_ico_nr_3" default="Icon Nr 3">
      <lang id="de" value="Icon Nr 3" />
      <lang id="en" value="Icon no 3" />
   </text>
</language>
```

GIRA

Gira
Giersiepen GmbH & Co. KG
Elektro-Installations-
Systeme

Industriegebiet Mermbach
Dahlienstraße
42477 Radvormwald

Postfach 1220
42461 Radvormwald

Deutschland

Tel +49 (0) 21 95-602-0
Fax +49 (0) 21 95-602-339

www.gira.de
info@gira.de