CE Customization

User Interface Extensions and Macros, CE9.4

With CE Customization you can add custom elements to your Touch10 operated video systems and DX Series user interface. Such user interface extensions may be in-room controls for lights or blinds, or other peripherals (including one or more video switches to extend the number of video sources available), all controlled by external control systems.

Macros allow you to write snippets of JavaScript code that can automate parts of your video endpoint behavior.

Since both the Cisco video system itself and peripherals now can be controlled from the Touch10/DX Series user interface, you will get a consistent user experience throughout the meeting room.

The current version of the CE Customization utility is available for the SX, MX, DX and Room Series video systems running Collaboration Endpoint Software, version CE9.4 or later.

Note that macros in the SX10 is not supported and that macros do not run on Webex enabled systems.

What's In This Document

On the Use of This Guide

When reading this on javascript enabled devices, the left menu bar and the entries in the table of contents are all hyperlinks. You can click on them to go to the topic.

Product Documentation

User guides, compliance and safety information for Cisco TelePresence systems are available at http://www.cisco.com/go/telepresence/docs

We recommend that you visit the Cisco web site regularly for updated versions of this guide.

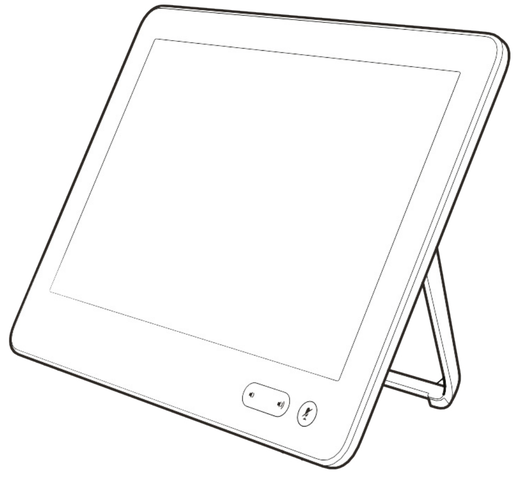
Who Has Access to the Editor?

In order to access the In-Room control editor you will need to have administrator rights.

However, an administrator may create an In-Room Control User account. With this account it is possible to log into the codec to run the In-Room Control Editor. No other part of the web interface is accessible from this account.

If you use SSH to log into the codec, only a very limited set of the API will be accessible.

Part 1
In-Room Control



Definition of Terms

The following terms will be used throughout this document:

Video system. Video system or codec in the Cisco TelePresence MX Series, SX or DX Series running Collaboration Endpoint Software, version CE9.3 or later. Sometimes referred to as video device.

Control system. Third-party control system with hardware drivers for peripherals, for example Crestron, AMX, Raspberry Pi.

Touch10. Our touch-based control device for the MX Series and SX Series video systems. Full product name: Cisco TelePresence Touch10. Also known as Touch10 controller, Touch10 user interface or Touch10 panel.

In-Room control panel. Panel with controls for third-party peripherals in the room. The panel opens when you tap the corresponding in-room control icon in the status bar on Touch10. See the Create a user interface chapter for more on this.

In-Room control editor. Our easy to use drag-and-drop editor for making in-room control panels.

xAPI. The bidirectional API of the video system. The xAPI allows third-party applications to interface with and interact with the video system, and vice versa.

Widget. User interface element, for example buttons, sliders, and text fields, that you can use to build an in-room control panel for Touch10.

With In-Room Control you can add custom elements to our Touch10 user interface. Such *user interface extensions* may be controls for lights or blinds, or other peripherals (including one or more video switches to extend the number of video sources available) all controlled by external control systems.

Since both the Cisco video system and the other peripherals now are controlled from the Touch10 user interface, you will get a consistent user experience throughout the meeting room.

The version of the In-Room Control feature described in this document, is available for the MX, SX (not the SX10) and DX Series video systems running Collaboration Endpoint Software, version CE9.3 or later.

In-Room Controls

You can customize the Touch10/DX user interface to allow control of peripherals in a meeting room, for example playback of a sound or movie source, lights and blinds.

You can also add content sensitive controls appearing only when in a call and/or only outside calls.

The maximum number of panels is now unlimited.**NEW**

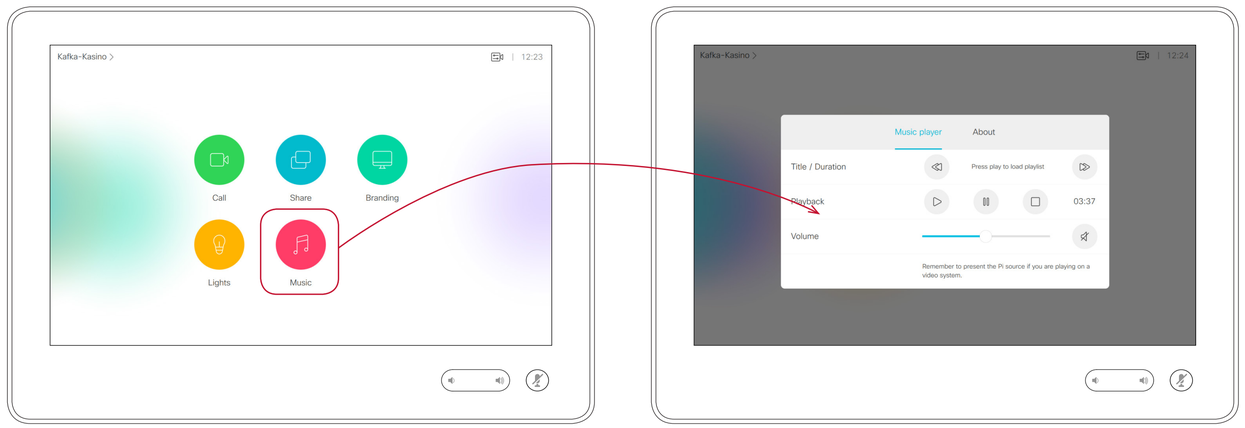
Observe that for all practical purposes the maximum number of panels will be set by usability requirements and, to some extent, the system resources. Each button you introduce on the Touch10/DX will need a corresponding panel.

All buttons will now appear along with the normal call handling buttons. Access to the Global buttons is no longer located in the top row of the Touch10/DX display.

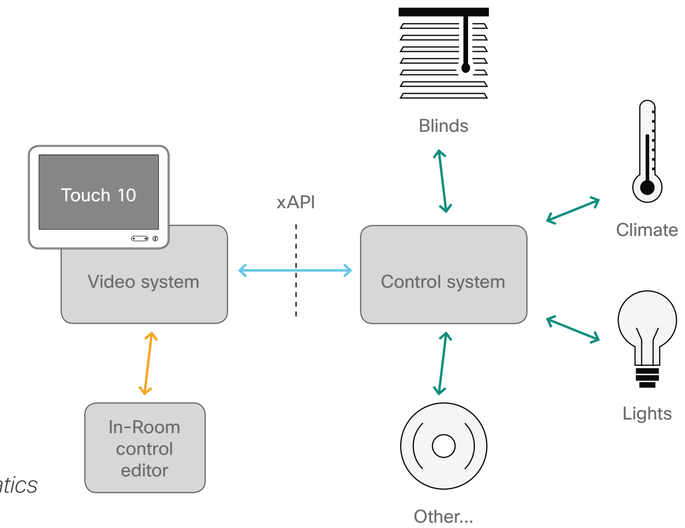
If there is not enough space left on the Touch10/DX display, a **More...** button will appear to provide access to the rest of the buttons.

This means that altogether you have three sets of panels at your disposal:

- **Always** icons (buttons), visible at all times.
- **Out of call only** icons (buttons), visible outside calls only.
- **In-Call only** icons (buttons), visible during calls only.



Examples of how customization made by means of In-Room Control may appear on the Touch10, with an icon as shown at left and the menu appearing when that icon has been tapped, allowing control of music player, as shown at right.



In-Room Control schematics

More About How In-Room Control Works

To utilize the features of the In-Room Control you will need a Cisco video system with a Touch10/DX user interface, and a third-party control system, for example Crestron, AMX.

The video system's API, referred to as the xAPI, is the link between the video system and the control system. Use the events and commands exposed by the xAPI when you program the control system.

The simple drag-and-drop editor offers a library of user interface elements, referred to as widgets. You can use these widgets to create your own in-room control panel for the Touch10/DX user interface.

Together, all of this provides a powerful combination of the control system's functionality and the user-friendly Touch10/DX user interface.

All examples in this document show Touch10 user cases only, but this should not cause difficulties due to the high degree of similarity between the two interfaces.

Messages

Trouble-shooting

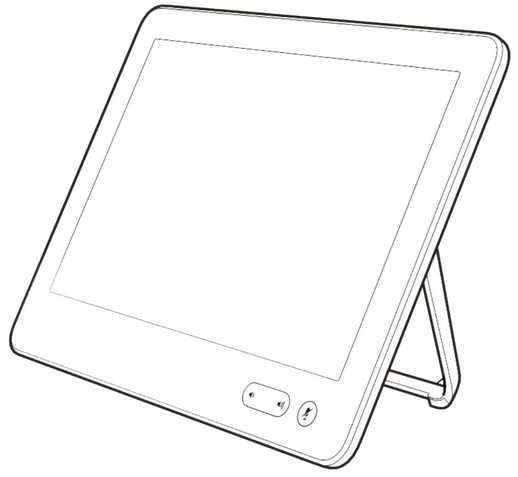
Tips & Tricks

Room Simulator

Video Switch

Macros

Creating a User Interface



Creating a User Interface for the Touch10

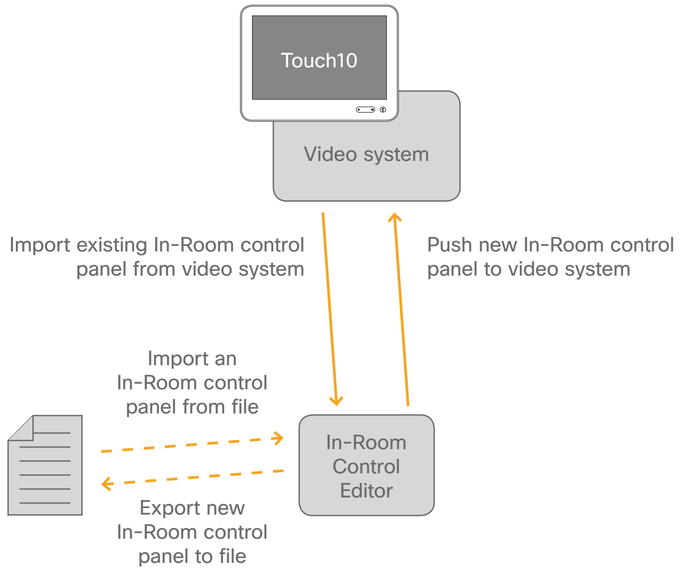
Use the In-Room Control Editor to create customized panels for peripheral controls on the video system's Touch10/DX user interface.

Connected to the Video System

If you have access to the video system, you can launch the editor from the video system's web interface.

If an in-room control panel already has been created on the Touch10/DX, this will automatically load into the editor, ready to act as a starting point for your design.

When you push a new panel to the video system, you will immediately see the result on the Touch10/DX.



In-Room Control Editor launched from the video system's web interface

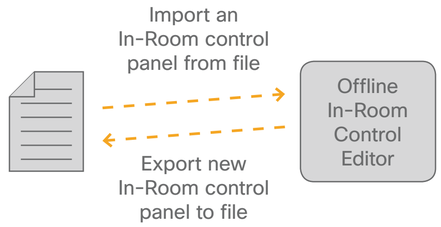
Offline

There are two places you can download the offline editor from:

- Download from http://www.cisco.com/go/in-room-control-docs
- Or, sign in to a video system's web interface with administrator credentials, navigate to **Integration** > **In-Room Control**, and click **Download Editor**.

If you choose to download the offline editor, extract the files from the downloaded zip-file. Retain the folder structure.

When using the offline editor you will be working with files, rather than communicating directly with the video system and Touch10/DX. Apart from this, the offline editor has full functionality.



Offline In-Room Control Editor

The editor that you launch from the video system's web interface and the offline editor share the same file format, so files created in one version can be opened and modified in the other.

Launch the In-Room Control Editor

Sign in to the video system's web interface with administrator credentials, navigate to **Integration** > **In-Room Control**, and click **Launch Editor**.

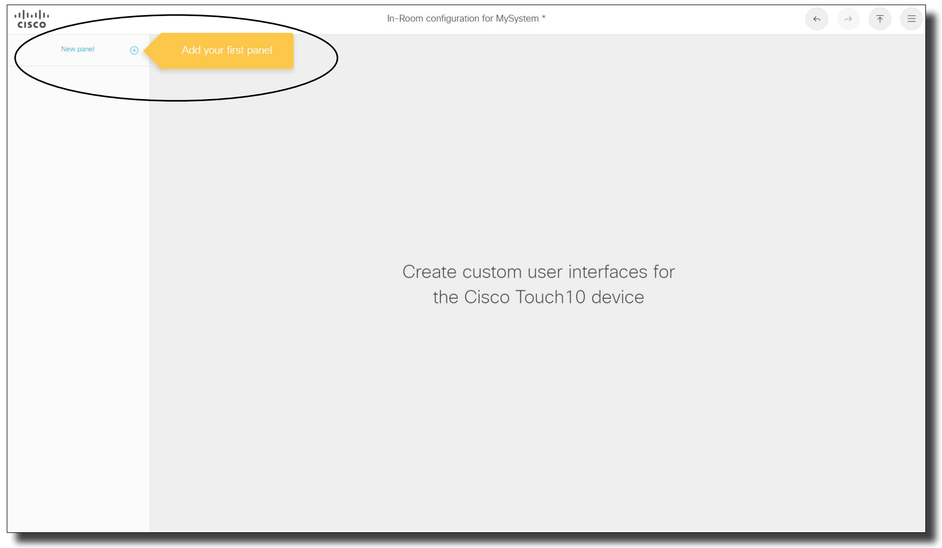
If there is a set of in-room control panels on the video system already, it will load automatically into the editor. The Create icons (as in the below example) will then read Edit wherever a panel has already been created.

Offline version. If you are using the offline version of the editor, use a browser to open the **index.html** file that you find in the **rceditor** folder.

With no panels already defined, the user interface will look as shown at right.

Click on ⊕ as indicated by the text **Add your first panel**.

The first panel will be of the **Home** panel type (see the previous pages for more on this). You can change this—see the next page for more.

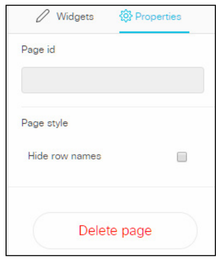


A Tour of the In-Room Control Editor

Use the Panel position (in Panel properties) to specify the position of a specific panel in the sequence (from top to bottom). The other panels will then be shifted accordingly. This will determine the order in which the buttons appear on the Touch10.

The Panel button name, color and icon appears here. To change the name, click on the text to make it editable. To specify color and icon use the Properties panel.

Create new panel from here.



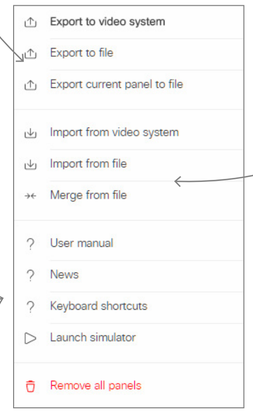
Page properties are accessed by clicking here.

Preview your current configuration, see also the following page.

Undo, Redo and Export configuration to video system.

More options

Entire sets of panels, or just a single panel can be exported to file for later use.

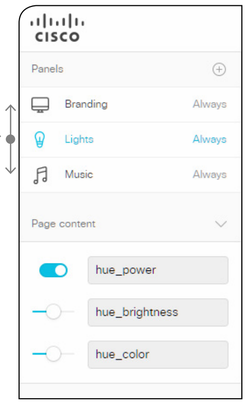


When importing from file, choose between Import and Merge. Merge will append panels to current set of panels. Any panels with the same name will then be overwritten.

ID is used to refer to a widget.

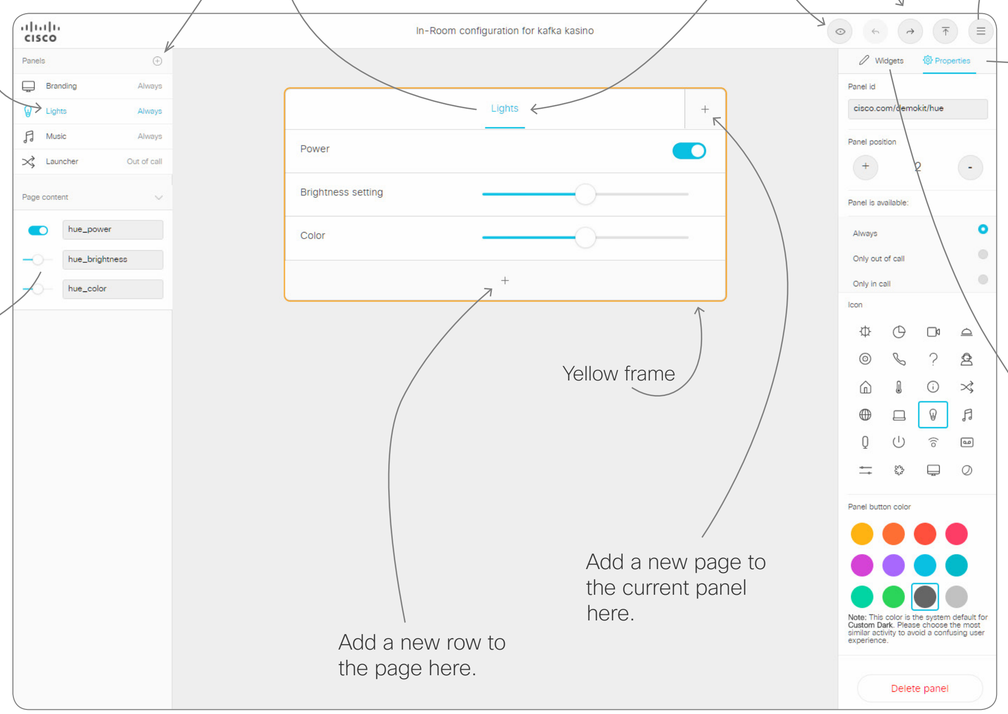
Text is what appears on the Touch10.

This controls the width of a widget. Width sizes available will depend on widget type.



The Widget IDs shown above belong to the active panel (indicated by text in blue).

Once you start to populate a page of a panel, the widget IDs in use will appear as shown above to provide a simpler overview.

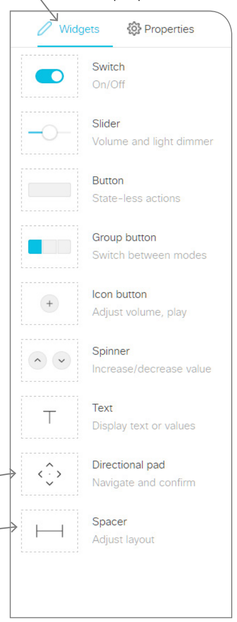


Panel properties

The properties panel will display settings for any part selected/highlighted by the yellow frame. Such a selection can be Panel, Page, Row or Widget.

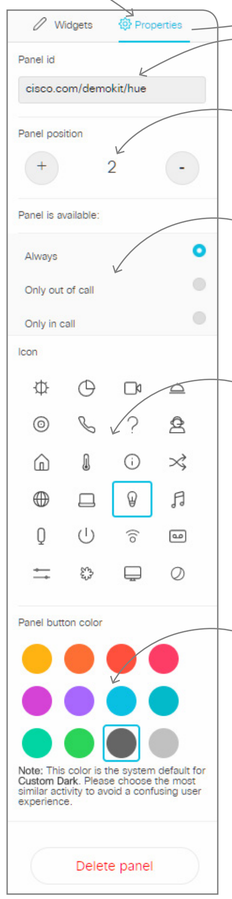
Tip! In addition to clicking the Undo and Redo icons, you may also use the familiar keyboard shortcut commands to copy and paste (Ctrl c & Ctrl v / Cmd c & Cmd v).

Drag widgets onto the page to populate it.



The spacer helps you get a better layout.

The Directional pad can be used to control e.g. Apple TV.



The panel ID.

Use this to change the order of the panels, see more about this in the text at top left.

Specify when the panel shall be available.

The icon chosen here will be the one that appears on the In-Room Control button for that panel on the Touch10/DX display.

You may specify the color of an In-room Control button appearing on the Touch10/DX screen.A limited color palette used for standard buttons is available in the editor. When you select a color, a small description of the context in which this color is used by Cisco, will be provided, as shown.

The maximum number of panels has now been increased to 20. A practical limit will be set by usability and, to some extent, the system resources. Each button you introduce on the Touch10/DX will need a corresponding panel. A panel will belong to one of the three following groups:

- In-call only (visible during calls only)
- Outside calls only (visible outside calls only)
- Always (visible at all times)

If you create more panels (i.e. buttons) than the Touch10 panel (or DX screen) can accommodate, a button called More will be created to give access to the excess buttons.

An in-room control panel can be arranged in pages. Each page consists of one or more rows, which you can populate with text and user interface elements known as widgets.

The maximum number of pages per panel is 50.

Widgets are arranged in a four-column grid. The widgets are placed into the grid according to the following rules:

- A widget fills between one and four columns depending on its size.
- Rows are right-aligned.
- If you add more widgets than fit in one line, widgets wrap to a new line within the same row.

Previewing Your Current Configuration

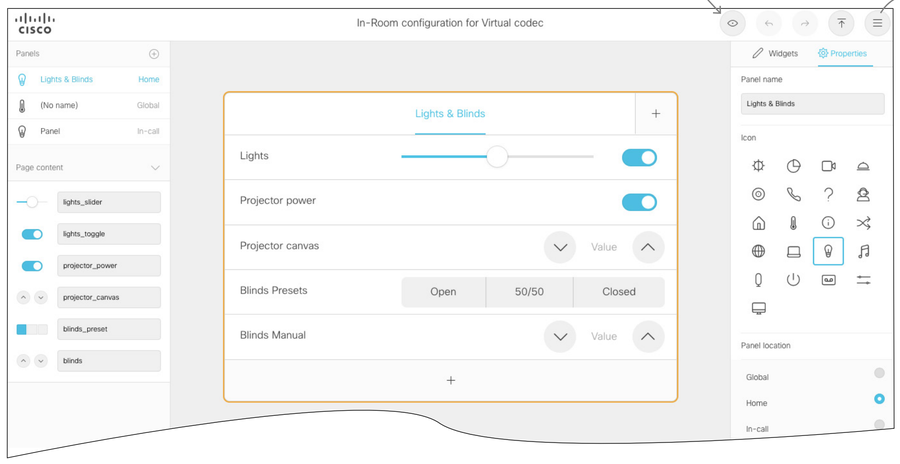
Messages
Trouble-shooting
Tips & Tricks
Room Simulator
Video Switch
Macros

You may preview your configurations to verify them before deploying them.

Note! In order to get a realistic preview of your configurations, you must launch the Simulator before you run the Preview described here. Otherwise the Touch10 will behave as if no control system is attached.

For the Preview to run successfully, just launch the Simulator. No further setup of the Simulator is needed. If the **Simulator config** has already been loaded into the Simulator, make sure you restore the Codec settings before you run the Preview. See “Running the Simulator” on page 42.

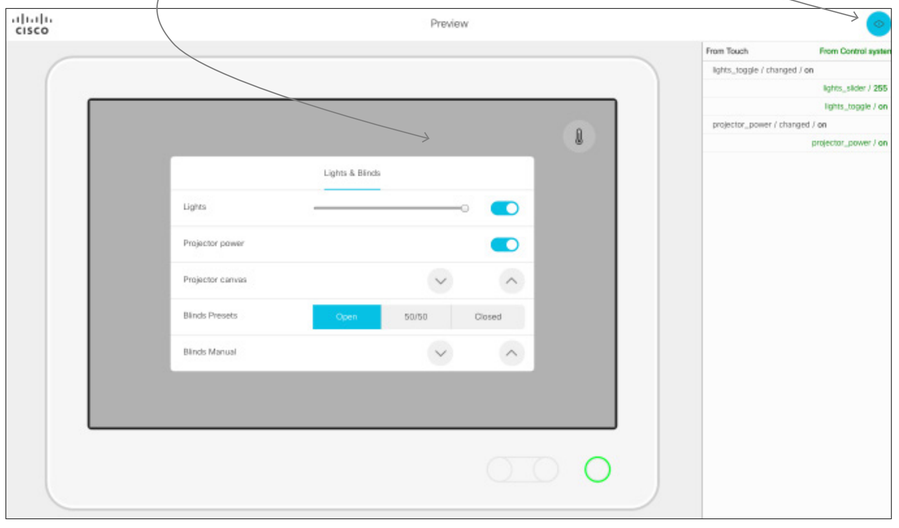
Click here to start the preview.



Simulator can be launched directly from the More (≡) menu.

Click here to return to the previous view.

Click in the image to change to displaying the buttons that will appear on the Touch10.



This pane shows the flow of information between a (simulated) control system and the Touch10.

The above provides a simulated view of your configuration, with a simulated third-party control system connected.

When implementing your configurations (a real situation scenario), make sure your control system has been set to send `SetValue` commands wherever applicable.

Example: If you set **Lights** to **On** in a real situation scenario, the Touch10 needs to receive feedback confirming that the lights actually are switched on. For this to take place, the controller must switch on the lights and then send a `SetValue`, confirming the change of the lights settings. The right pane of the above example shows a simulation of what the Touch10 sends to the Control system and what the Control system then sends back to the Touch10.

In a real situation scenario, you should also make sure that the control system sends a `SetValue` to the Touch10 whenever someone operates the light switch on the wall in the meeting room.

Note! If you did not activate the Simulator before running this preview, the configuration will act as if no `SetValue` commands are received by the Touch10. When no such feedback is received, the **Lights** button on the Touch10 will return to its **Off** setting within a short while.

For more on this, see the following pages.

Messages

Trouble-shooting

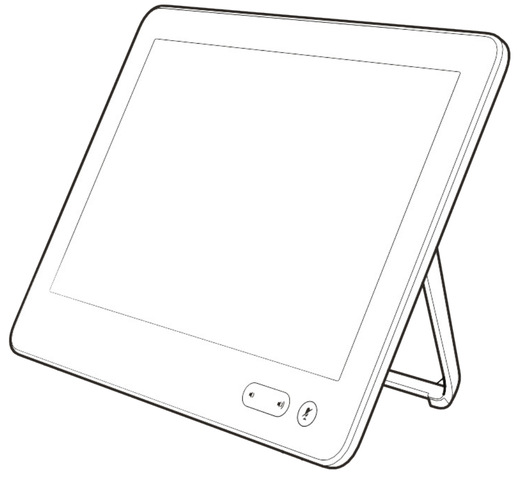
Tips & Tricks

Room Simulator

Video Switch

Macros

Application Programming Interface (API)



API for Programming In-Room Controls

Connect to the Video System

The video system's API (also known as the xAPI) allows bidirectional communication with third-party control systems, such as those from AMX or Crestron. There are multiple ways to access the xAPI:

- Telnet
- SSH
- HTTP/HTTPS
- RS-232/serial connection

Regardless of the method you choose, the structure of the xAPI is the same. Choose the access method that suits your application and video system the best.

Consult the API guide for your video system to get a full description of available access methods and how to use the xAPI.

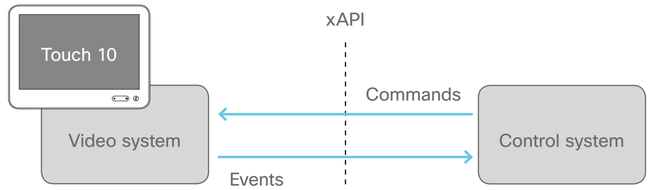
Go to:
http://www.cisco.com/go/sx-docs for SX Series
http://www.cisco.com/go/mx-docs for MX Series
http://www.cisco.com/go/dx-docs for DX Series

Then, click **Reference Guides** > **Command References** to find the API guides.

Communicate over the API

The video system and the control system exchange messages through the xAPI to make sure that the Touch10/DX In-Room Control panel always reflects the actual status of the room.

The video system sends one or more events when someone uses one of the controls on the Touch10/DX In-Room Control panel, and the control system should send a command to the video system when there is a change in the room settings.



The video system and the control system exchange messages through the xAPI.

Examples:

- When someone taps a **Lights On** button on Touch10/DX, the video system sends the associated events. The control system should respond to these events by switching on the lights in the room and send the corresponding command back to the video system.
- When someone switches on the lights in the room, the control system should send a command to the video system, so that the video system can update the Touch10/DX In-Room Control panel to reflect that the light is on.

See the Command reference chapter for an overview of all relevant events, commands and statuses for in-room control.

Pairing Video System and Control System

You can register the control system as a peripheral connected to the video system:

```
xCommand Peripherals Connect ID: "ID" Type: ControlSystem
```

where ID is the unique ID for the control system, typically the MAC address.

See the API guide for more details about this command, and its options.

Heartbeats. The control system must send heartbeats to the video system to let the video system know that the control system is connected. The control system stays on the connected devices list (refer to xStatus Peripherals ConnectedDevice) as long as the video system receives these heartbeats from the control system.

```
xCommand Peripherals HeartBeat ID: "ID" [Timeout: Timeout]
```

where ID is the unique ID for the control system, typically the MAC address, and Timeout is the number of seconds between each heartbeat. If Timeout is unspecified, it is assumed to be 60 seconds.

Note. If a connected unit ceases to send heartbeats, some time will elapse until the video system detects the absence of heartbeats—as long as up to a couple of minutes.

This works the other way around as well, up to a couple of minutes may elapse until new heartbeats are detected by the codec.

API for Programming In-Room Controls (Cont.)

Events for Widget Actions

The video system sends one or more of the following events when someone uses the controls on the Touch10/DX in-room control panel:

- **Pressed**—sent when a widget is first pressed
- **Changed**—sent when changing a widget's value (applies to toggle buttons and sliders only)
- **Released**—sent when a widget is released (also when moving away from the widget before releasing)
- **Clicked**—sent when a widget is clicked (pressed and released without moving away from the widget).

These events are sent in two versions:

- **UserInterface Extensions Event**—suited for *terminal output mode*
- **UserInterface Extensions Widget**—suited for *XML output mode.*

See the table at right to find out the version best suited for your control system to register to.

When, and by which widgets (user interface elements), these events are triggered, are described in the Widgets chapter.

UserInterface Extensions Event (suited for terminal output mode)	**UserInterface Extensions Widget** (suited for XML output mode)
A single string contains information about the type of action, which widget triggered the event (identified by the Widget ID), and the widget value.	The type of action, which widget triggered the event (identified by the Widget ID), and the widget value are included as separate elements in the XML tree.

How to register:
`xfeedback register event/UserInterface/Extensions/Event`
Example:
`*e UserInterface Extensions Event Pressed Signal: "WidgetId:Value"`
`** end`
`*e UserInterface Extensions Event Changed Signal: "WidgetId:Value"`
`** end`
`*e UserInterface Extensions Event Released Signal: "WidgetId:Value"`
`** end`
`*e UserInterface Extensions Event Clicked Signal: "WidgetId:Value"`
`** end`
	How to register:
	`xfeedback register event/UserInterface/Extensions/Widget`
	Example:
	`<Event>`
	`<UserInterface item="1">`
	`<Extensions item="1">`
	`<Widget item="1">`
	`<Action item="1">`
	`<WidgetId item="1">WidgetId</WidgetId>`
	`<Value item="1">Value</Value>`
	`<Type item="1">Type</Type>`
	`</Action>`
	`</Widget>`
	`</Extensions>`
	`</UserInterface>`
	`</Event>`

Two event versions that a control system can register to: one suited for terminal output mode, the other for XML output mode

API for Programming In-Room Controls (Cont.)

Event for Panel Update

The video system sends the following event when a new In-Room Control panel is applied:

LayoutUpdated—sent when a new in-room control panel for Touch10/10 is exported to the video system.

As a response to this event, the control system should send commands to initialize all widgets so that they reflect the true status of the room settings.

How to register:

```
xfeedback register event/UserInterface/Extensions/Widget/
   LayoutUpdated
```

Example:

Terminal output mode:

```
*e UserInterface Extensions Widget LayoutUpdated
** end
```

XML output mode:

```
<Event>
  <UserInterface item="1">
    <Extensions item="1">
      <Widget item="1">
        <LayoutUpdated item="1"/>
      </Widget>
    </Extensions>
  </UserInterface>
</Event>
```

Event for Opening or Closing of a Page

If you have given each of your pages a unique Page ID, the system can send events when a page is opened or closed.

EventPageOpened—sent when a page is opened
EventPageClosed—sent when a page is closed

The pages are like radio buttons, opening another page will close the current page. In that case both the EventPageClosed and the EventPageOpened will be issued.

How to register:

```
xfeedback register event/UserInterface/Extensions/PageOpened
xfeedback register event/UserInterface/Extensions/PageClosed
```

Example:

Terminal output mode:

```
*e UserInterface Extensions Event PageOpened PageId:
   "appletvpage"
*e UserInterface Extensions Event PageClosed PageId:
   "appletvpage"
```

XML output mode:

```
<Event>
  <UserInterface item="1">
    <Extensions item="1">
      <Page item="1">
        <Action item="1">
          <PageId item="1">appletvpage</PageId>
          <Type item="1">Opened</Type>
        </Action>
      </Page>
    </Extensions>
  </UserInterface>
</Event>
```

For an example of PageClosed, just substitute Closed for Opened in the example at left. This event will typically be used when you want the controller to take some action based on the event, in this case turning on (off) the AppleTV box for you.

API for Programming In-Room Controls (Cont.)

Commands and Statuses

The SetValue command, which sets the value of a widget, is essential when working with in-room controls:

```
xCommand UserInterface Extensions Widget SetValue
   Value: Value WidgetId: WidgetId
```

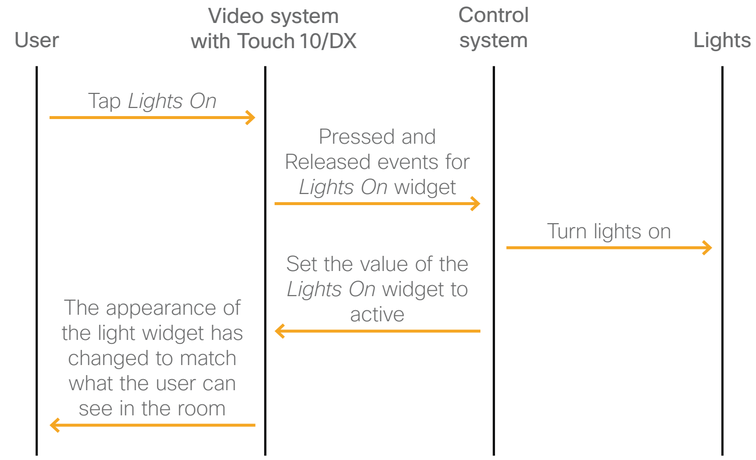
When the video system receives a SetValue command, the video system's status and the Touch10/DX In-Room Control panel are updated accordingly.

It is important that the control system sends `SetValue` commands in the following situations, so that the Touch10/DX In-Room Control panel truly reflects the status of the room:

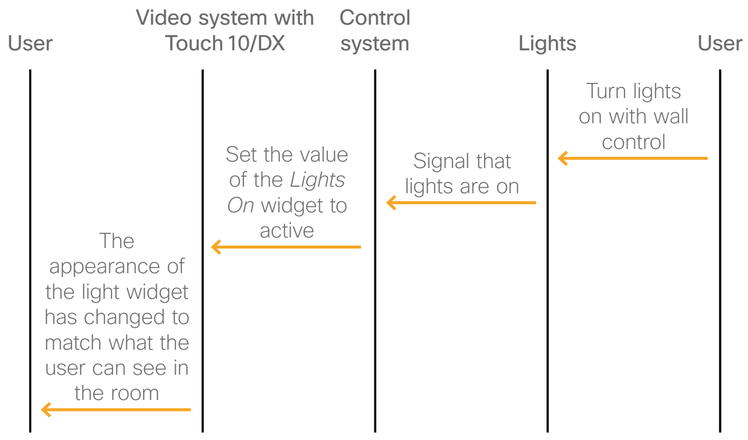
- When the control system initially connects to the video system.
- When the video system restarts.
- When the control system restarts.
- When a new In-Room Control panel is exported to the video system from the In-Room Control editor (as response to the LayoutUpdated event).
- When someone physically changes something in the room, for example turns on the lights using a wall control.
- As a response to an event, for example when someone has tapped the *Lights On* button on the Touch10/DX In-Room Control panel.
- The control system must also do all that is necessary in the room to reflect the action on the Touch10/DX In-Room Control panel, for example actually switch on the light.

Consult the Widgets chapter for more details about which commands apply to the different widgets (user interface elements).

Examples



Message flow–turn on the lights using the controls on Touch10/DX



Message flow–turn on the lights using the wall control

Messages

Trouble-shooting

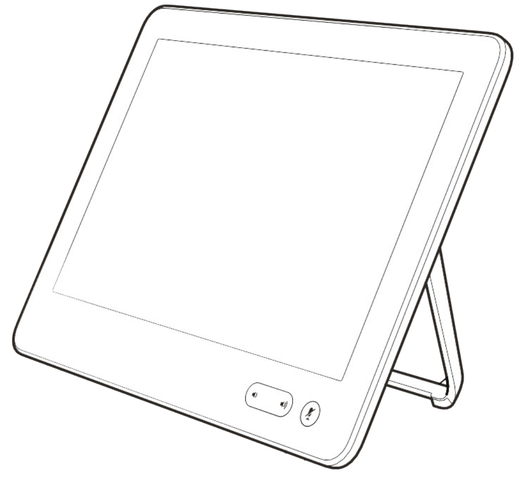
Tips & Tricks

Room Simulator

Video Switch

Macros

Widgets



Overview of Widgets

About Widgets

The Touch10/DX In-Room Control panel is composed of user interface elements called widgets. You can find the complete widget library in the right pane of the In-Room Control editor.

General tab: Buttons with custom text, group buttons, toggle button, sliders, text fields and more.

Icons tab: Buttons with familiar symbols for Home, Power, Arrow up/down/left/right, Camera controls, Loudspeaker controls, Microphone control, Media player controls, and more.

Each of the widget type available are described on the following pages, with emphasis on:

- Commands that change the value of the widget
- Events that are sent (pressed, changed, released, clicked) and which actions trigger these events
- Examples of commands and events, both in terminal output mode and XML output mode.

Syntax and semantics for all events, commands and statuses that are related to in-room controls (user interface extensions) are included in the Command reference chapter.

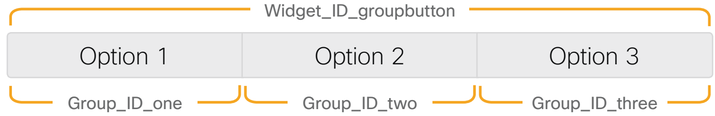
The Widget Identifier

All widgets on a Touch10/DX in-room control panel need a unique identifier, a Widget ID. The Widget ID may either be defined by you, or assigned automatically. The Widget ID can be any name or number; we recommend using a descriptive name without special characters. The maximum number of characters is 255.

The Widget ID is the programming link between Touch10/DX, the video system, and the control system. The Widget ID will be included in all events that are associated with a widget, and you must use the same identifier when you send commands to that widget via the code that you write for your control system.

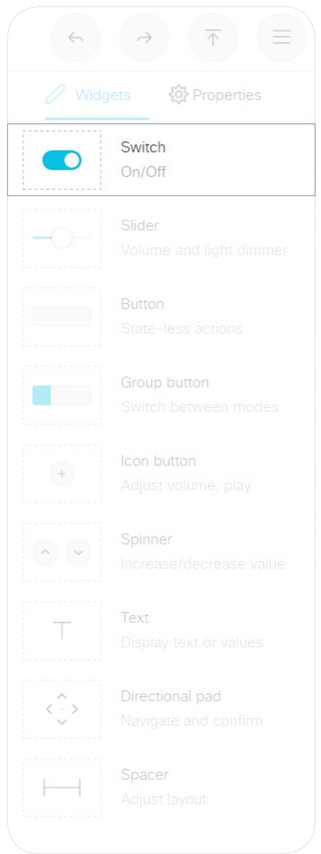
Group Identifiers

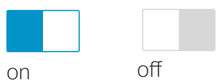
One of the widgets, the Group button, has two types of identifiers: The Widget ID refers to the complete group of buttons, while Group IDs are unique identifiers for the individual buttons within the group.



A Group ID is assigned automatically, but can be defined by you instead. A Group ID can be any name or number; we recommend using a descriptive name without special characters. The maximum number of characters is 255.

Switch





A two-state switch which indicates either on or off.

Events

Changed—triggered when the button is released.

Value: <on/off>

Example: Press “on" on a switch with WidgetId = " togglebutton".

Terminal mode

```
*e UserInterface Extensions Event Changed Signal: "togglebutton:on"
** end
```

XML mode

```
<Event>
  <UserInterface item="1">
    <Extensions item="1">
      <Widget item="1">
        <Action item="1">
          <WidgetId item="1">togglebutton</WidgetId>
          <Value item="1">on</Value>
          <Type item="1">changed</Type>
        </Action>
      </Widget>
    </Extensions>
  </UserInterface>
</Event>
```

Commands

The visual appearance of the button changes immediately when you tap it. However, the control system must always send a `SetValue` command to the video system when the button toggles between *on* and *off*. This ensures that the status is updated accordingly.

Example: Set a button with WidgetId = " togglebutton" to " on".

```
xCommand UserInterface Extensions Widget SetValue WidgetId: "togglebutton" Value: "on"
```

Switch is a two-state button which indicates either on or off.

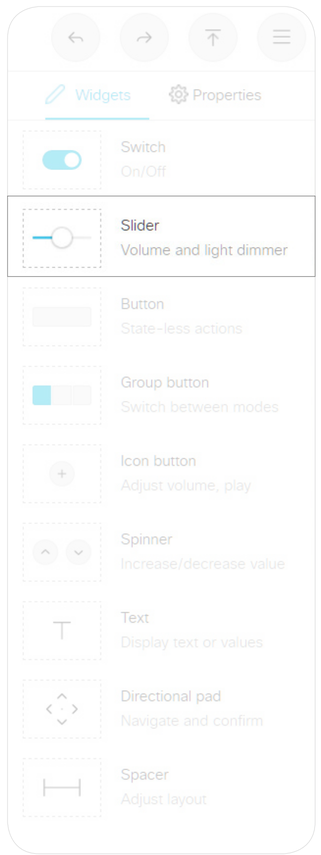
Example of use: Anything that can be turned on or off, for example lights, fan, and projector.

You can also use it as a toggle button together with a slider for lights to be dimmed.

Messages
Trouble-shooting
Tips & Tricks
Room Simulator
Video Switch
Macros

Slider

Search
About
UI
API
W WIDGETS
CMD REFERENCE
Messages
Trouble-shooting
Tips & Tricks
Room Simulator
Video Switch
Macros



Events

Pressed — Triggered when the slider is pressed
Value: N/A

Changed — Triggered when the slider is moved while holding down, and when the slider is released.
Value: 0-255

Released — Triggered when the slider is released
Value: 0-255

Example: Press the slider with WidgetId = “slider”, and move it into a new position (“68”), and release.

Terminal mode

```
*e UserInterface Extensions Event Pressed Signal: “slider”
** end
*e UserInterface Extensions Event Changed Signal: “slider:32”
** end
*e UserInterface Extensions Event Changed Signal: “slider:68”
** end
*e UserInterface Extensions Event Released Signal: “slider:68”
** end
```

XML mode

```
<Event>
  <UserInterface item=”1”>
    <Extensions item=”1”>
      <Widget item=”1”>
        <Action item=”1”>
          <WidgetId item=”1”>slider</WidgetId>
          <Value item=”1”>68</Value>
          <Type item=”1”>released</Type>
        </Action>
      </Widget>
    </Extensions>
  </UserInterface>
</Event>
```

Commands

The visual appearance of the slider changes immediately when you tap or slide it. However, the control system must always send a SetValue command to the video system to tell the new position of the slider. This ensures that the status is updated accordingly.

Example: Set the slider with WidgetId = “slider” to position “98”.

```
xCommand UserInterface Extensions Widget SetValue WidgetId: “slider” Value: “98”
```

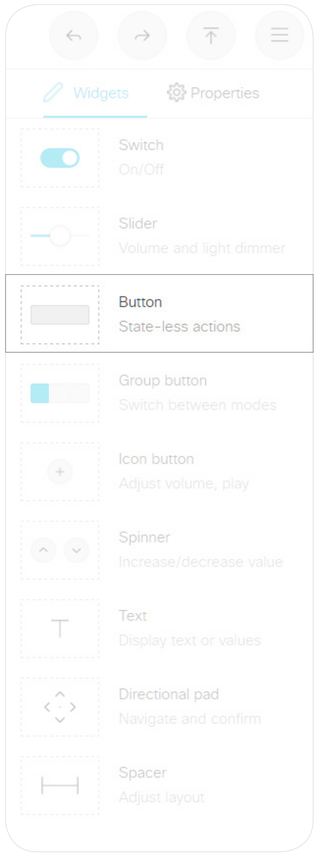
About Sliders

A slider selects values within a set range. The minimum value is represented by 0, and the maximum value is represented by 255. When the slider is being pressed and moved, events are sent maximum 5 times a second.

When you tap the bar, the slider is immediately moved to that new position.

Example of use: Dimmable lights, volume control.

Button



Events

Pressed	Triggered when the button is pressed. Value: N/A
Changed	Triggered when the button is released. Value: N/A
Released	Triggered when the button is released. Value: N/A

Example: Press and release the button with WidgetId = “button”.

Terminal mode

```
*e UserInterface Extensions Event Pressed Signal: “button”
** end
*e UserInterface Extensions Event Released Signal: “button”
** end
*e UserInterface Extensions Event Clicked Signal: “button”
** end
```

XML mode

```
Event>
  <UserInterface item=”1”>
    <Extensions item=”1”>
      <Widget item=”1”>
        <Action item=”1”>
          <WidgetId item=”1”>button</WidgetId>
          <Value item=”1”></Value>
          <Type item=”1”>clicked</Type>
        </Action>
      </Widget>
    </Extensions>
  </UserInterface>
</Event>
```

Commands

Use the `SetValue` command to highlight or not the button in the user interface. A value of “active” will highlight the button, and a value of “inactive” will release it.

Example: Highlight the button with WidgetId = “button” (set it in active state).

```
xCommand UserInterface Extensions Widget SetValue WidgetId: “button” Value: “active”
```

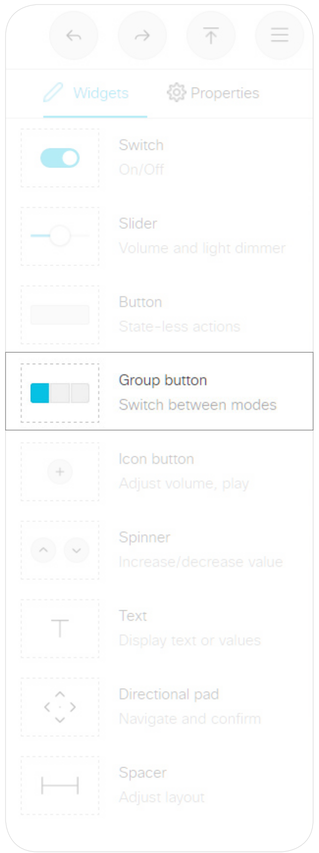
Buttons with custom text come in different sizes. The size determines the maximum number of characters you can add. Text does not wrap to a new line. You cannot use the SetValue command to change the text dynamically.

A button has two states: active and inactive. You do not have to set the button in active state when someone taps it; the button can be used to just send a signal without changing the button’s visual state.

If you want to have the buttons linked so that only one can be selected at a time (radio buttons), consider to use Group buttons (next page).

Example of use: Switching things on and off.

Group Button



Events

Pressed — Triggered when one of the buttons is pressed. **Value:** The Group ID of the button (within the group) that was pressed.

Released — Triggered when one of the buttons is released. **Value:** The Group ID of the button (within the group) that was released

Example: There are four buttons in the group with WidgetId = “groupbutton”. Press the button with Group ID = “two”.

Terminal mode

```
*e UserInterface Extensions Event Pressed Signal: “groupbutton:two”
** end
*e UserInterface Extensions Event Released Signal: “groupbutton:two”
** end
```

XML mode

```
<Event>
  <UserInterface item="1">
    <Extensions item="1">
      <Widget item="1">
        <Action item="1">
          <WidgetId item="1">groupbutton</WidgetId>
          <Value item="1">two</Value>
          <Type item="1">released</Type>
        </Action>
      </Widget>
    </Extensions>
  </UserInterface>
</Event>
```

Commands

The visual appearance of the button changes immediately when you tap it. However, the control system must always send a SetValue command to the video system when one of the buttons are tapped. This ensures that the status is updated accordingly.

Use the `UnSetValue` command to release all buttons in the group so that no button is highlighted.

Example: Select (highlight) the button with Group ID = “one” in the group with WidgetId = “groupbutton”. Then, release all buttons (no buttons are highlighted).

```
xCommand UserInterface Extensions Widget SetValue
   WidgetId: “groupbutton” Value: “one”

xCommand UserInterface Extensions Widget UnsetValue
   WidgetId: “groupbutton”
```

Group buttons may now be made as a matrix and not just as a line.[NEW] This means that you are no longer confined to a maximum of 4 radio buttons.

A matrix consists of up to 4 columns and as many rows as you need.

You start by defining how many columns your matrix should consist of (1, 2, 3 or 4). This is a global setting applying to the entire matrix (i.e. all the rows) and it defines the maximum number of buttons per row.

However, a row may contain fewer buttons than this maximum number. Button autosizing willl then take place—the buttons will always fill the space available.

Example: Assume that you have defined the matrix to consist of 3 columns and you need 7 buttons (i.e. 3 rows). The system will then put 3 in the first row and 3 in the second row, and the last button in the third row. The single button in the third row will be autosized to fill the space (spanning all 3 columns).

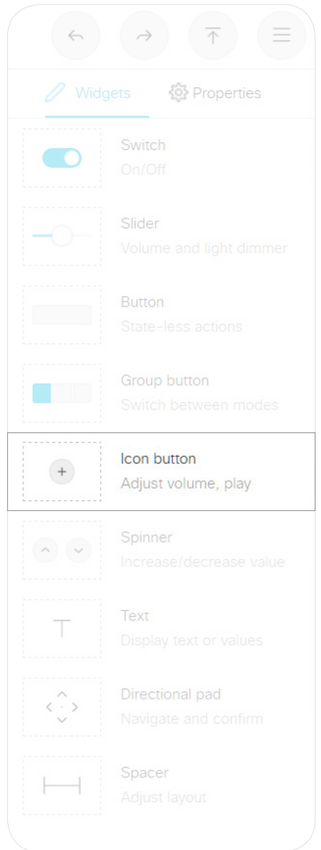
The size of a button determines the maximum number of characters you can add. Text does not wrap to a new line, but willl be truncated, whenever needed.

You cannot use the `SetValue` command to change the text dynamically.

Example of use: Room presets that are mutually excluding, like room presets where you can choose between Dark, Cool, and Bright. Remember to deselect (release) the preset, if it is no longer valid (for instance when changing the lights with a wall control, or a Touch 10 slider.

Another example of use: Changing to a different UI language.

Icon Button



Events

Pressed	Triggered when the button is pressed. Value: N/A
Released	Triggered when the button is released.Value: N/A
Clicked	Triggered when the button is released Value: N/A

Example: Press and release the button with WidgetId = “symbol”.

Terminal mode

```
*e UserInterface Extensions Event Pressed Signal: “symbol”
** end
*e UserInterface Extensions Event Released Signal: “symbol”
** end
*e UserInterface Extensions Event Clicked Signal: “symbol”
** end
```

XML mode

```
<Event>
  <UserInterface item="1">
    <Extensions item="1">
      <Widget item="1">
        <Action item="1">
          <WidgetId item="1">symbol</WidgetId>
          <Value item="1"></Value>
          <Type item="1">clicked</Type>
        </Action>
      </Widget>
    </Extensions>
  </UserInterface>
</Event>
```

Commands

Use the `SetValue` command to highlight or not the button in the user interface. A value of “`active`” will highlight the button, and a value of “`inactive`” will release it.

Example: Highlight the button with WidgetId = “symbol” (set it in active state)

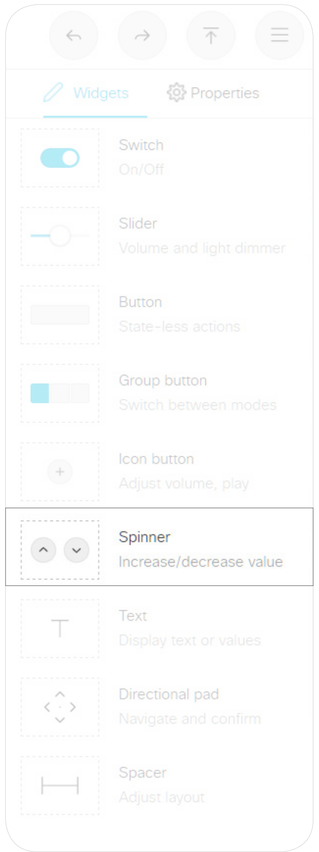
```
xCommand UserInterface Extensions Widget SetValue WidgetId: “symbol”
   Value: “active”
```

Icon buttons share behavior with buttons having custom text.

A button has two states: active and inactive. You do not have to set the button in active state when someone taps it; the button can be used to just send a signal without changing its visual state.

Example of use: Controls for a media player, or other devices that can start, stop, pause.

Spinner



Events

Pressed	Triggered when one of the spinner buttons is pressed. **Value:** <increment/decrement>
Released	Triggered when one of the spinner buttons is released. **Value:** <increment/decrement>
Clicked	Triggered when one of the spinner buttons is released. **Value:** <increment/decrement>

Example: Press and release the decrement button of the spinner with WidgetId = “spinner”.

Terminal mode

```
*e UserInterface Extensions Event Pressed Signal: “spinner:decrement”
** end
*e UserInterface Extensions Event Released Signal: “spinner:decrement”
** end
*e UserInterface Extensions Event Clicked Signal: “spinner:decrement”
** end
```

XML mode

```
<Event>
  <UserInterface item=”1”>
    <Extensions item=”1”>
      <Widget item=”1”>
        <Action item=”1”>
          <WidgetId item=”1”>spinner</WidgetId>
          <Value item=”1”>decrement</Value>
          <Type item=”1”>clicked</Type>
        </Action>
      </Widget>
    </Extensions>
  </UserInterface>
</Event>
```

Commands

Use the `SetValue` command to add or update the text between the two buttons.

Example: For the spinner with WidgetId = “spinner”, add the text “22” between the spinner’s increment and decrement buttons.

```
xCommand UserInterface Extensions Widget SetValue WidgetId: “spinner”
   Value: “22”
```

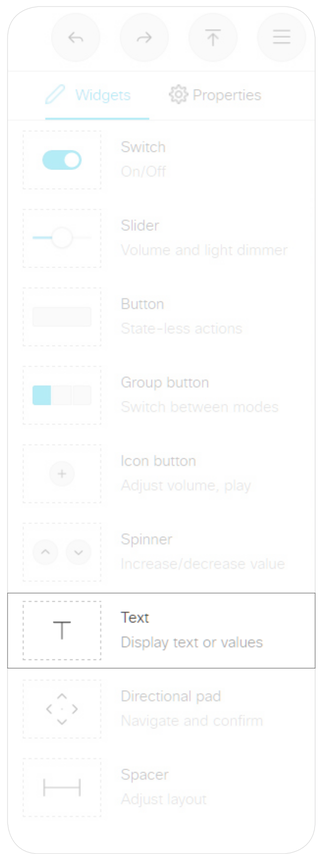
A spinner is used to step through a list of values. You may use the two buttons to increment or decrement a number, or to step through a list of options.

Use the `SetValue` command to add text between the buttons.

Example of use: Set the desired temperature in the room.

Messages
Trouble-shooting
Tips & Tricks
Room Simulator
Video Switch
Macros

Text



Events

None

Commands

Use the `SetValue` command to set the text in the text box.

Example: Set the following text in the text box with WidgetId = “textbox”: “The projector is warming up.”.

```
xCommand UserInterface Extensions Widget SetValue WidgetId: “textbox”
   Value: “The projector is warming up.”
```

Text boxes come in different sizes. They have up to two lines of text and the text automatically wraps to the new line.

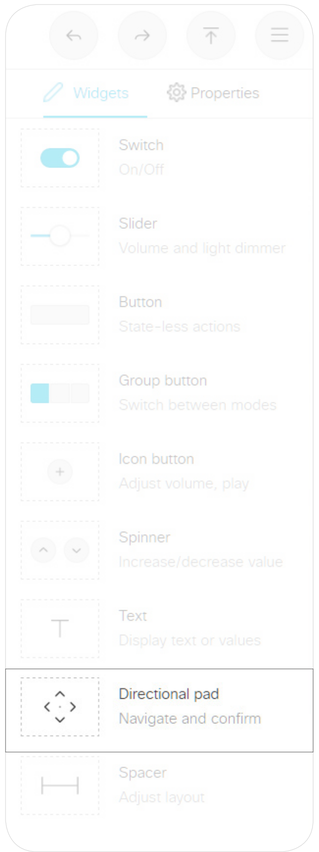
A small text box with larger font size and no line wrap is also available.

You can define the initial text for the text box in the editor, and later on use the SetValue command to enter text dynamically.

Example of use: Help text, instructions, explanation of what different presets mean, or informative text from the control system, such as “The projector is warming up”.

The text box with larger font size is primarily meant for status values, such as the current temperature in the room.

Directional Pad



Events

Pressed	Triggered when the button is pressed. Value: N/A
Changed	Triggered when the button is released. Value: N/A
Released	Triggered when the button is released. Value: N/A

Example: Press and release the button with WidgetId = “dirpad”.

Terminal mode

```
*e UserInterface Extensions Event Pressed Signal: “dirpad:up”
** end
*e UserInterface Extensions Event Released Signal: “dirpad:up”
** end
*e UserInterface Extensions Event Clicked Signal: “dirpad:up”
** end
```

XML mode

```
<Event>
  <UserInterface item=”1”>
    <Extensions item=”1”>
      <Widget item=”1”>
        <Action item=”1”>
          <WidgetId item=”1”>dirpad</WidgetId>
          <Value item=”up”></Value>
          <Type item=”1”>clicked</Type>
        </Action>
      </Widget>
    </Extensions>
  </UserInterface>
</Event>
```

The Directional Pad can be regarded as a set of 5 buttons, the four Directional buttons and the Center button.

As can be seen from the examples at left, the event will be of the form:

“<WidgetId>:<*the button pushed*>”

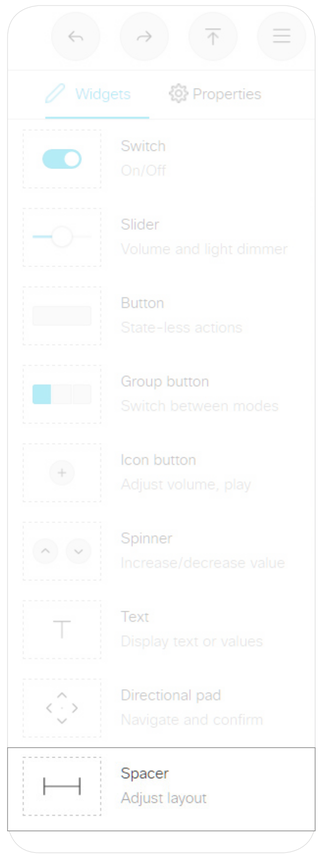
in which *the button pushed* assumes the value:

up, *down*, *left*, *right* or *center*

Example of use: Controlling AppleTV

Messages
Trouble-shooting
Tips & Tricks
Room Simulator
Video Switch
Macros

Spacer



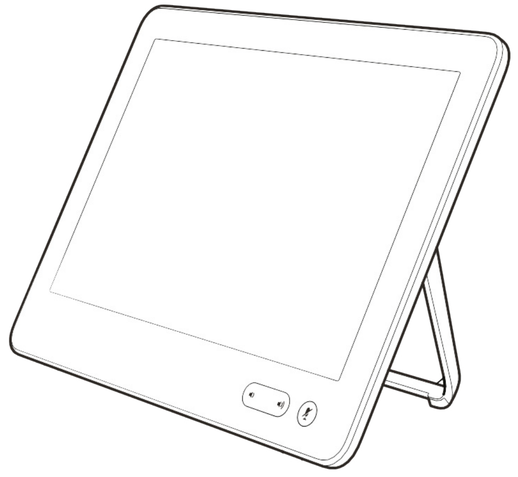
The Spacer is no more than a layout tool. Consequently, there are no events or commands associated with it.

The Spacer lets you add space between or after widgets. It is no more than a layout tool.

The width of the spacer is adjustable (1–4). If you set it to maximum it will occupy its own line, making it usable as a vertical spacer, as well.

Messages
Trouble-shooting
Tips & Tricks
Room Simulator
Video Switch
Macros

Command Reference



Events

UserInterface Extensions Event Pressed

Sent by the video system when a widget is first pressed.

Equivalent to the UserInterface Extensions Widget Action event with Type "Pressed".

```
*e UserInterface Extensions Event Pressed Signal: Signal
```

in which

Signal: String (0, 255)

The format of the string is "<WidgetId>:<Value>", where <WidgetId> is the unique identifier of the widget that triggers the event, and <Value> is the value of the widget. The range of allowed values depends on the widget type.

UserInterface Extensions Event Released

Sent by the video system when a widget is released (even if moving the finger out of the widget before releasing it).

Equivalent to the UserInterface Extensions Widget Action event with Type "Released".

```
*e UserInterface Extensions Event Released Signal: Signal
```

in which

Signal: String (0, 255)

The format of the string is "<WidgetId>:<Value>", where <WidgetId> is the unique identifier of the widget that triggers the event, and <Value> is the value of the widget. The range of allowed values depends on the widget type.

UserInterface Extensions Event Changed

Sent by the video system when changing a widget's value (applies only to toggle buttons and sliders).

Equivalent to the UserInterface Extensions Widget Action event with Type "Changed".

```
*e UserInterface Extensions Event Changed Signal: Signal
```

in which

Signal: String (0, 255)

The format of the string is "<WidgetId>:<Value>", where <WidgetId> is the unique identifier of the widget that triggers the event, and <Value> is the value of the widget. The range of allowed values depends on the widget type.

UserInterface Extensions Event Clicked

Sent by the video system when a widget is clicked (pressed and released without moving the finger out of the widget).

Equivalent to the UserInterface Extensions Widget Action event with Type "Clicked".

```
*e UserInterface Extensions Event Clicked Signal: Signal
```

in which

Signal: String (0, 255)

The format of the string is "<WidgetId>:<Value>", where <WidgetId> is the unique identifier of the widget that triggers the event, and <Value> is the value of the widget. The range of allowed values depends on the widget type.

Events (Cont.)

UserInterface Extensions Widget Action

Sent by the video system when someone uses one of the controls on the user interface (in-room control panel).

Equivalent to the UserInterface Extensions Event *Type* event.

Depending on the action type, this event is equivalent to one of these events:

- UserInterface Extensions Event Pressed
- UserInterface Extensions Event Changed
- UserInterface Extensions Event Released
- UserInterface Extensions Event Clicked Events

```
<Event>
  <UserInterface item="1">
    <Extensions item="1">
      <Widget item="1">
        <Action item="1">
          <WidgetId item="1">WidgetId</WidgetId>
          <Value item="1">Value</Value>
          <Type item="1">Type</Type>
        </Action>
      </Widget>
    </Extensions>
  </UserInterface>
</Event>
```

in which:

WidgetId: String (0, 40)
The unique identifier for the widget that triggered the event.

Value: String (0, 255)
The value of the widget. The range of allowed values depends on the widget type.

Type: <Pressed/Changed/Released/Clicked>

Pressed: Sent when a widget is first pressed.

Changed: Sent when changing a widget's value (only for toggle buttons and sliders).

Released: Sent when a widget is released (even if moving the finger out of the widget before releasing it).

Clicked: Sent when a widget is clicked (pressed and released without moving the finger out of the widget).

UserInterface Extensions Widget LayoutUpdated

Sent by the video system when the configuration file for the user interface extensions has been updated, i.e. when exporting a new configuration from the in-room control editor to the video system.

```
*e UserInterface Extensions Widget LayoutUpdated
```

or

```
<Event>
  <UserInterface item="1">
    <Extensions item="1">
      <Widget item="1">
        <LayoutUpdated item="1"/>
      </Widget>
    </Extensions>
  </UserInterface>
</Event>
```

Commands

UserInterface Extensions Widget SetValue

This command sets the value of the given widget, and the UserInterface Extensions statuses are updated accordingly. If the value is out of range, the command returns an error.

USAGE:

```
xCommand UserInterface Extensions Widget SetValue Value: Value
    WidgetId: WidgetId
```

in which

Value: String (0, 255)
The value of the widget. The range of values depends on the widget type.

WidgetId: String (0, 40)
The unique identifier for the widget.

UserInterface Extensions Widget UnsetValue

This command empties the value of the given widget, and the UserInterface Extensions statuses are updated accordingly. The user interface is notified that the widget is no longer selected.

USAGE:

```
xCommand UserInterface Extensions Widget UnsetValue WidgetId:
    WidgetId
```

in which

WidgetId: String (0, 40)
The unique identifier for the widget.

UserInterface Extensions Clear

This command deletes all user interface extensions (widgets) from the video system.

USAGE:

```
xCommand UserInterface Extensions Clear
```

UserInterface Extensions List

Use this command to list all user interface extensions (widgets) that exist on the video system.

USAGE:

```
xCommand UserInterface Extensions List
```

Statuses

UserInterface Extensions Widget [n] WidgetId

UserInterface Extensions Widget [n] Value

This status returns the identifier (WidgetId) and the current value of the widgets.

The value is an empty string until a value is set by using the UserInterface Extensions Widget SetValue command.

USAGE:

```
xStatus UserInterface Extensions
```

Value space of the result returned:

Value: The value of the widget. Depends on widget type. String (0, 255).

WidgetId: The unique widget identifier. String (0, 40).

Example:

```
xstatus UserInterface Extensions
*s UserInterface Extensions Widget 1 Value: “on”
*s UserInterface Extensions Widget 1 WidgetId: “togglebutton”
*s UserInterface Extensions Widget 2 Value: “255”
*s UserInterface Extensions Widget 2 WidgetId: “slider”
*s UserInterface Extensions Widget 3 Value: “Blinds”
*s UserInterface Extensions Widget 3 WidgetId: “spinner”
*s UserInterface Extensions Widget 4 Value: “inactive”
*s UserInterface Extensions Widget 4 WidgetId: “button”
*s UserInterface Extensions Widget 5 Value: “2”
*s UserInterface Extensions Widget 5 WidgetId: “groupbutton”
*s UserInterface Extensions Widget 6 Value: “Projector is
   ready”
*s UserInterface Extensions Widget 6 WidgetId: “textfield”
** end
```

Messages

Trouble-shooting

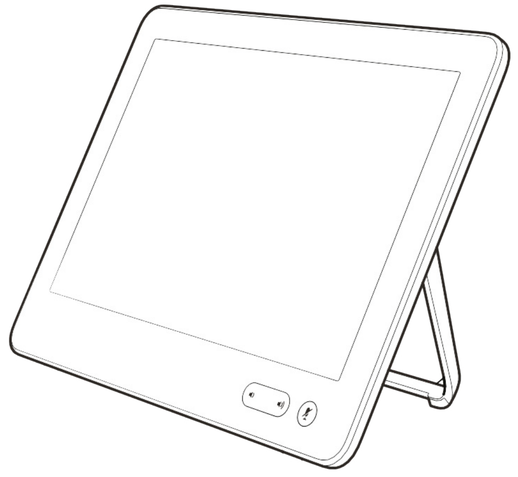
Tips & Tricks

Room Simulator

Video Switch

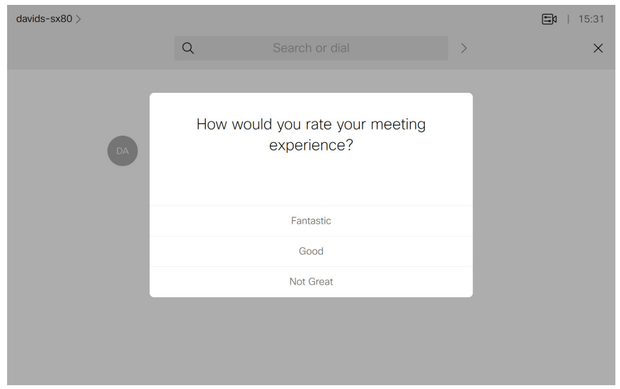
Macros

Creating Interactive Messages



How Interactive Messages Work (I)

Example 1—Rating Your Experience

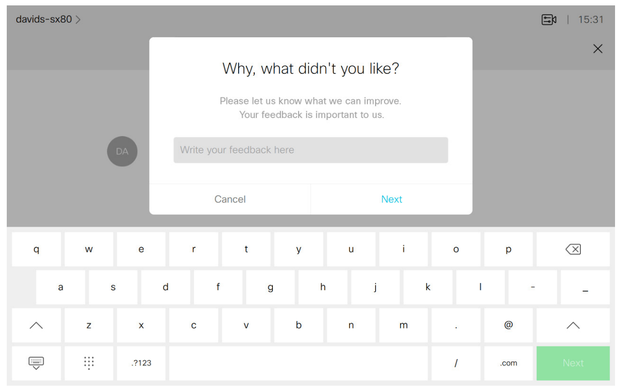



```
xCommand UserInterface Message Prompt Display
   FeedbackId: “MeetingExperience” Text: “” Title:
   “How would you rate your meeting experience?”
   Option.1: “Fantastic” Option.2: “Good” Option.3:
   “Not Great”


*e UserInterface Message Prompt Response FeedbackId:
   “MeetingExperience”
*e UserInterface Message Prompt Response OptionId: 1


<XmlDoc resultId=””>
<Event>
  <UserInterface item=”1”>
    <Message item=”1”>
      <Prompt item=”1”>
        <Response item=”1”>
          <FeedbackId item=”1”>MeetingExperience</
   FeedbackId>
          <OptionId item=”1”>1</OptionId>
        </Response>
      </Prompt>
    </Message>
  </UserInterface>
</Event>
</XmlDoc>
```

Example 2—Write Your Feedback Here




```
xCommand UserInterface Message TextInput Display
   FeedbackId: “MeetingFeedback” Placeholder: “Write
   your feedback here” SubmitText: “Next” Title: “Why,
   what didn’t you like?” Text: “Please let us know
   what we can improve. Your feedback is important to
   us.”


*e UserInterface Message TextInput Response
   FeedbackId: “MeetingFeedback”
*e UserInterface Message TextInput Response Text: “Low
   resolution”


<XmlDoc resultId=””>
<Event>
  <UserInterface item=”1”>
    <Message item=”1”>
      <TextInput item=”1”>
        <Response item=”1”>
          <FeedbackId item=”1”>MeetingFeedback</
   FeedbackId>
          <Text item=”1”>Low resolution</Text>
        </Response>
      </TextInput>
    </Message>
  </UserInterface>
</Event>
</XmlDoc>
```

The Messages feature lets you create alerting and/or interactive messages on the Touch10/DX screen prompting the user to act accordingly.

If you want to create a sequence of messages where the next message depends on a choice made in the previous message, we recommend the use of macros to create events to act upon. Alternatively, you may use an external control device, which then will act upon the events created.

In order to submit inputs from the user, you should make use of HttpFeedback.

The HttpFeedback enables you to get the device to post http feedback messages (also known as webhooks) upon changes to the API state, e.g. statuses, events and configuration updates. The HTTP Post feedback messages will be sent to the specified ServerURL.

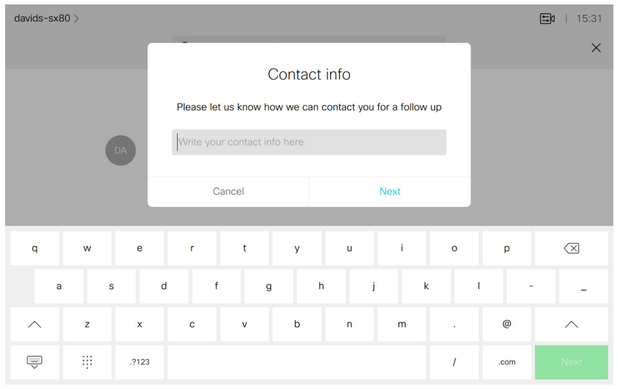
More about this can be found in the API Reference Guide.

Note that when you create messages as shown in example 2, at left, you may specify the text on the “Next” button. The “Cancel” button, however, appears by default and its text cannot be altered.

When you create messages using Message Alert as in example 4 (on the next page), the “Dismiss” button will appear by default. The text on this button cannot be altered.

How Interactive Messages Work (II)

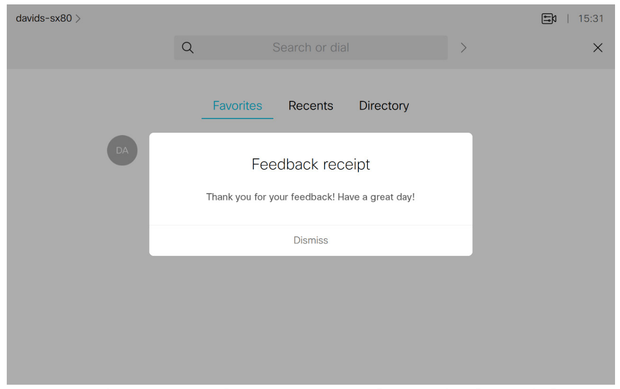
Example 3—Getting In Touch With You




```
xCommand UserInterface Message TextInput Display
   FeedbackId: “ContactInfo” Placeholder: “Write
   your contact info here” SubmitText: “Next” Title:
   “Contact Info” Text: “Please let us know how we
   can contact you for a follow up”


*e UserInterface Message TextInput Response
   FeedbackId: “ContactInfo”
*e UserInterface Message TextInput Response Text:
   “john@go.webex.com”
<XmlDoc resultId=””>
<Event>
  <UserInterface item=”1”>
    <Message item=”1”>
      <TextInput item=”1”>
        <Response item=”1”>
          <FeedbackId item=”1”>ContactInfo</
   FeedbackId>
          <Text item=”1”>john@go.webex.com</Text>
        </Response>
      </TextInput>
    </Message>
  </UserInterface>
</Event>
</XmlDoc>
```

Example 4—Feedback Receipt

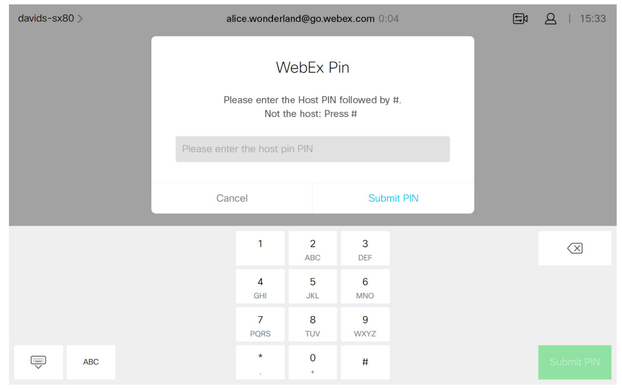



```
xCommand UserInterface Message Alert Display Title:
   “Feedback receipt” text: “Thank you for your
   feedback! Have a great day!”


*e UserInterface Message Alert Cleared


<XmlDoc resultId=””>
<Event>
  <UserInterface item=”1”>
    <Message item=”1”>
      <Alert item=”1”>
        <Cleared item=”1”/>
      </Alert>
    </Message>
  </UserInterface>
</Event>
</XmlDoc>
```

Example 5—Enter Your WebEx Pin




```
xCommand UserInterface Message TextInput Display
   FeedbackId: “WebExPin” InputType: Numeric
   Placeholder: “Please enter the host pin PIN”
   SubmitText: “Submit PIN” Text: “Please enter the
   host pin PIN, followed by #. Not the host: Press #”
   Title: “WebEx Pin”


*e UserInterface Message TextInput Response
   FeedbackId: “WebExPin”
*e UserInterface Message TextInput Response Text:
   “1122#”


<XmlDoc resultId=””>
<Event>
  <UserInterface item=”1”>
    <Message item=”1”>
      <TextInput item=”1”>
        <Response item=”1”>
          <FeedbackId item=”1”>WebExPin</FeedbackId>
          <Text item=”1”>1122#</Text>
        </Response>
      </TextInput>
    </Message>
  </UserInterface>
</Event>
</XmlDoc>
```

Messages

Trouble-shooting

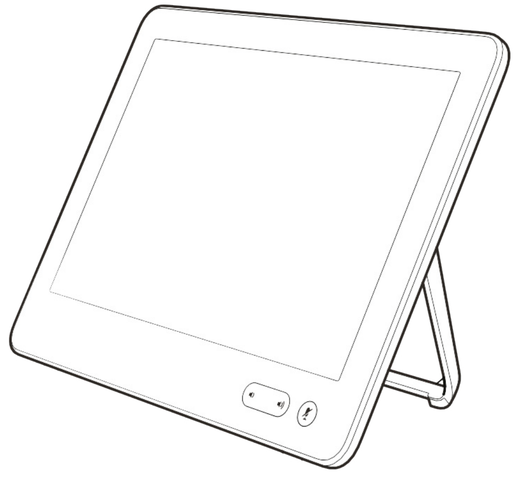
Tips & Tricks

Room Simulator

Video Switch

Macros

Troubleshooting



Tips When Troubleshooting

Sign In

Sign in to the video system's web interface with administrator credentials, navigate to **Integration** > **In-Room Control**. Click the arrow to show the **Development Tools**.

Overview of all Widgets and Their Status

The **Widget State Overview** window lists all widgets, and their status. The status is shown in the **Current Value** column.

If the **Current Value** column is empty, the widget has not been initialized and has no value. We recommend that the control system initializes all widgets when it initially connects to the video system.

Send Value Updates to the Video System

A control system sends `SetValue` commands to the video system, telling it to update a widget. For test purposes, you can use the **Update Value** column in the **Widget State Overview** window to simulate a control system.

Enter a value in one of the input fields to immediately send the corresponding `SetValue` command to the video system. The **CurrentValue** column (status) will be updated, and the Touch10 in-room control panel changes accordingly.

Click **Unset** to clear the value of the widget (send an UnsetValue command).

If a control system is connected to the video system, the **Current Value** and **Update Value** columns may come out-of-sync. The **Current Value** column always shows the current status, regardless of whether the `SetValue` command is sent from a real control system, or from the **Update Value** column.

Check for Events and Status Updates

All events and status updates related to widgets appear immediately in the **Log** window. Events are prefixed with *e, and statuses are prefixed with *s.

Events appear when you use the controls on the Touch 10 user interface, and the status is updated when a command, which changes the video system's status, is sent to the video system.

If a Panel Fails to Load

If an existing in-room control panel failed to load automatically on launching the editor, you may manually import the panel(s) from codec or load a local file made with the offline editor.

All alternatives erase any unsaved data in the editor, but the existing in-room control panel on the video system is neither overwritten nor deleted until a new panel is exported to the video system.

Make Sure That Macros Are Not the Cause

If you experience unintended behavioral changes and you run macros on your system, make sure you deactivate the macros before proceeding with your troubleshooting.

Use `xConfiguration Macros Mode: On/Off` to do this.

The macro framework has its own log file called **macros.log**

The **macros.log** file contains much of what is printed in the Macro console. The macros can be configured to print output to the console and this will be stored in the log, so keep in mind that you can see custom log messages (which must have been created by the developer) in this file.

Messages

Trouble-shooting

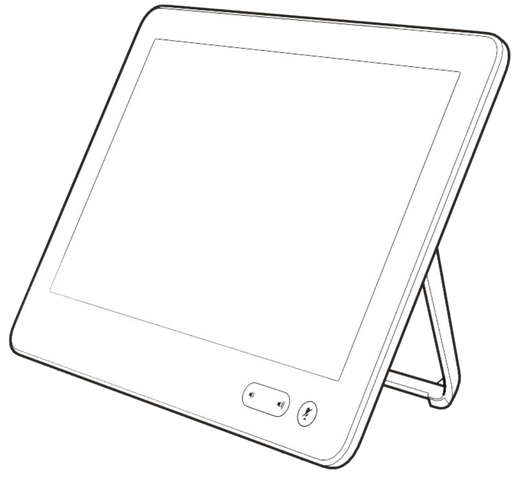
Tips & Tricks

Room Simulator

Video Switch

Macros

Tips and Tricks



Recommended Best Practice

Re-register to Get Feedback After Restart

When either the video system or the control system restarts, the control system must re-register to the events that the video system sends when someone uses the Touch10/DX in-room controls or pushes a new in-room control panel to the Touch10/DX.

For terminal output mode:

```
xfeedback register event/UserInterface/Extensions/Widget
```

For XML output mode:

```
xfeedback register event/UserInterface/Extensions/Event
```

```
xfeedback register event/UserInterface/Extensions/Widget/
   LayoutUpdated
```

Consult the API for in-room control chapter for more details.

Initialize All Widgets

Make sure the control system initializes all the widgets on the Touch10/DX in-room control panel in the following situations:

- When the control system connects to the video system for the first time
- When the video system restarts
- When the control system restarts
- When a new in-room control panel is exported to the video system (as response to a LayoutUpdated event).

If this is not done, then the Touch10/DX may show incorrect values and not truly reflect the status of the room.

Use the `SetValue` command to set the initial values.

Always send values back to the video system when something changes.

To avoid unexpected behavior and ambiguities, the control system must always send `SetValue` commands to the video system when something changes. This applies also when the change is triggered by someone using the controls on the Touch10/DX.

For example, it makes no difference if you use a slider on the Touch10/DX in-control panel to dim the light, or a physical dimmer in the room, or another touch panel. The control system must always send the dimmer value back to the video system using the `SetValue` command.

Update the In-Room Control Panel

When you export a new in-room control panel to the video system, the old panel is overwritten and replaced by the new one.

To update, do as follows:

1. Launch the In-Room Control Editor from the video system's web interface.
2. Create the In-Room Control panel you want, or import a previously saved panel from file (**Import** > **From file**).
3. Click **Export** > **To codec**.

Other Useful Stuff

Remember previous values when turning lights off (e.g. a light with a slider for dimming and toggle for on/off), remember the current light state when turning off, and use that value when turning back on.

Example: If the light is at 40%, and the user presses off, she will expect it to go back to 40% (not 100%) when pressing the "on" again. Remember also to set the power switch to off if sliding to 0%.

For blinds, consider using the following strategy: Short press on a **Direction** arrow tilts the blinds. If tilted all the way already, a short press moves the blinds slightly.

Long press on a **Direction** arrow starts moving the blind that direction, doesn't stop until blinds are all the way up/down.

To stop movement after long pressing, short press any button (stop button is not really necessary)

Remove Entire In-Room Control Panel and Icon

If there is an In-Room Control panel on the video system then there is also a corresponding In-Room Control icon, either in the Touch10/DX status bar or as a button to the right of the call control buttons. Even if a panel is empty and contains no widgets, both the icon and the panel will be visible.

Do as follows to remove an In-Room Control panel and icon from Touch10/DX:

1. Launch the In-Room Control editor from the video system's web interface.
2. Select the panel to be removed (Global, Homescreen or In-call)
3. Click **Delete panel**.

Transition From Third-party Control Systems to CE

If you already have been using a third party control system and want to start using CE as described in this document, we recommend the following:

- Let any programming made to control third party stuff remain untouched.
- Remove all code that controls the Cisco video endpoint as that is now already controlled via the Touch10/DX.
- Reprogram the signaling from the button-presses coming from the third party control system panel so that it listens to button presses from the Touch10/DX instead.

This programming can be very simple to do as the largest control system manufacturers provide modules/drivers for In-Room Controls making it very easy to get started with the programming.

Recommended Best Practice (cont.)

Inspirational Examples

The following examples may serve as inspiration and to provide further guidance on best practices. It is by no means mandatory to design and implement controls as illustrated in these examples.

Widget ID

When you drag a widget (e.g. a button) onto a page, you may give it a customized ID. Widget IDs do not have to be unique. Widgets can share ID, but they must be of the same type. This means that you can have two sliders in different panels called "main-light", but you cannot have one slider and one toggle button both called "main-light".

To create a duplicate of an existing widget on another page or panel, just use Copy and Paste.

Create Groups of What Belongs Together

Consider grouping controls that belong together on the same page. The pages you create in the in-room control editor appear as separate tabs on the control panel.

Control of Lights

The combination of a slider and a toggle button could be used to control lights. The toggle button switches the lights on or off; the slider serves as a dimmer.

Consider the following strategy:

- Set a slider to minimum when the user turns the lights off.
- Set a toggle button to off when the user moves the slider to its minimum.
- Remember the value of the slider when the lights are turned off, so that you can return to this value when the lights are turned back on again.
- If the light is at 40 %, when the user switches it off, he or she would expect it to go back to 40 % (not maximum) when switching the lights on again.
- When the user selects one of the options in the group button (a light preset), set the sliders and toggle buttons accordingly.
- If the lights are changed away from a preset, for instance by changing a slider or toggle button, deselect all options in the group button.

Control of Temperature

The combination of a spinner and a large font text box (value) may be used to control temperature. Use the spinner to set the desired temperature, and the large font text box to show the current temperature.

For the best user experience keep the following in mind:

- Update the large font text box as the temperature in the room changes. Update the text field of the spinner when someone tap the up and down arrows
- Consult the Widgets chapter for details about how to update the spinner's text field and the large font text box.

Control of Blinds

You can either use a spinner, or up and down arrows from the Icons tab in the widget library.

Consider the following strategy:

- Tilt the slides as response to a short press on a direction arrow. If tilted all the way, move the blinds up or down incrementally.
- As response to a long press on a direction arrow, start moving the blinds in that direction. They do not stop until all the way up or down.
- Short press any button in order to stop the movement after a long press. Then no separate stop button will be needed.

Group Buttons

Group buttons (radio buttons) are ideal when you want buttons to be linked, so that only one can be selected at a time. For example room presets. When the individual buttons in a group are too small to contain the text that describes their function, consider to use text boxes for the description.

Create Speed Dial or One Button to Push

If you want to create buttons on the Touch10 that directly result in an action without displaying any kind of panels or settings, do as follows:

- Create a new panel and then click *Delete page*, so that the panel contains no pages at all.

When you tap on the button on the Touch10, this will directly create an event, which in turn can be used to start an action. Typical examples of use of this feature is Speed Dialing or One Button to Push solutions.

Make sure you give the Touch10 button a descriptive name for the user.

Granting Access to In-Room Control Editor and Extensions API

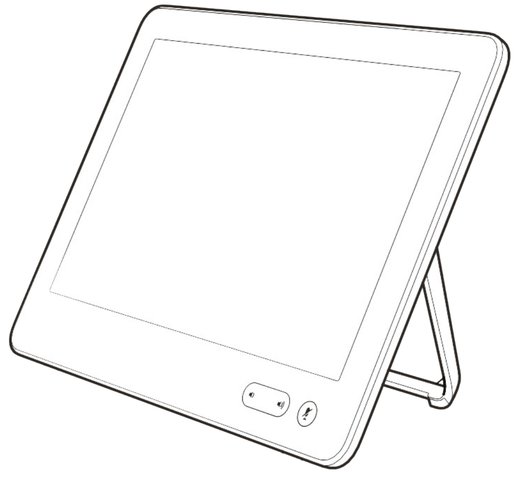
In order to access the In-Room Control Editor you will need to have administrator rights.

However, an administrator may create an In-Room Control User account. With this account it is possible to log into the codec to run the In-Room Control Editor. No other part of the web interface is accessible from this account.

If you use SSH to log into the codec, only a very limited set of the API will be accessible.

Messages
Trouble-shooting
Tips & Tricks
Room Simulator
Video Switch
Macros

Part 2
Room Simulator



Running the Simulator

In order to make it possible to demonstrate the features of the In-Room Control, we have created a Simulator for you. The purpose of the Simulator is two-fold. It may serve as a sales tool, but it also proves useful if you want to study how commands and events are sent back and forth.

The Simulator contains its own simulated third-party control system.

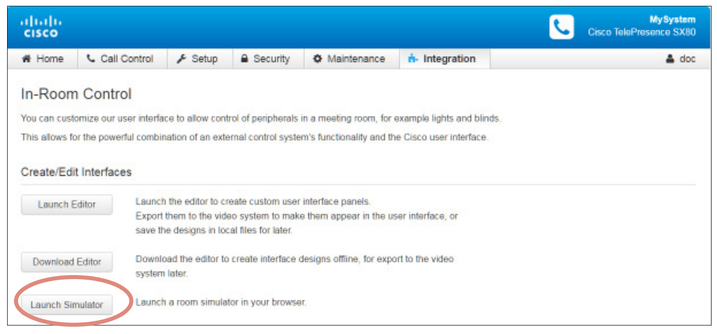
Used as a Sales Tool

When used as a sales tool to show what this is all about, the Simulator uses a predefined In-Room Control setup. This cannot be changed. The predefined setup will overwrite the setup in the codec. Your current setup will be backed up in your browser and reinstalled after the session (you will be prompted to confirm that).

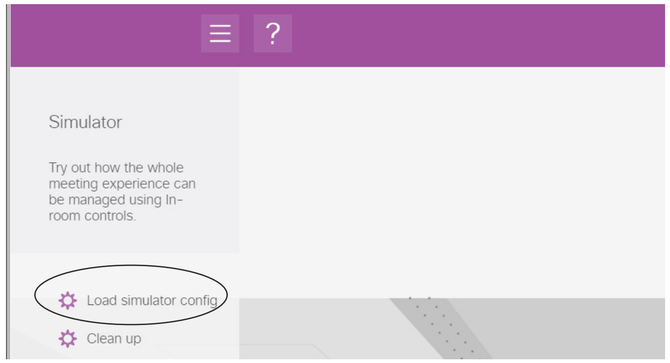
Note! As the Simulator is self-contained, any controller connected to the codec should be disconnected when running the Simulator to avoid conflicts.

Tip! We recommend using Google Chrome browser when running the Simulator. Other browsers may fail to run the Simulator properly.

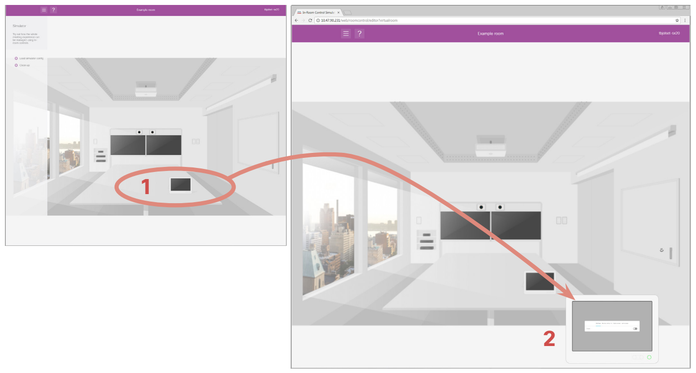
To invoke the Simulator navigate to **Integration** > **In-Room Control** in the web interface and then select **Launch Simulator**:



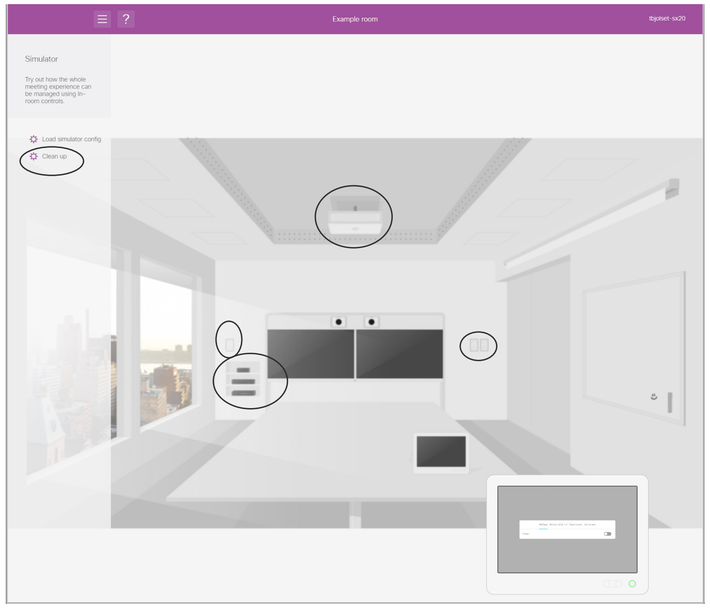
The Example Room will now appear. Click on **Load simulator config**. Once successfully loaded click anywhere to remove messages when needed.



Click on the Touch10 (1) to enlarge it and then click as shown (2) to invoke the In-Room Controls:



You may now play with the controls on the virtual Touch10, but also with the switches themselves in the room. Click **Clean up** to finish and restore previous settings.



Used as a Preview Simulator

You may also use the simulator to preview the effects of your configuration and to monitor the information interchange between the Touch10 and a simulated third-party control system. Make sure that you have **not** loaded the simulator config into the Simulator. Restore the Codec config, if needed (use **Clean up** as discussed above).

How to use the Preview Simulator is described in “Previewing Your Current Configuration” on page 10.

Part 3
Use of a Video Switch

Using a Third-party Video Switch to Extend the Number of Video Sources Available

The Cisco Touch10 panel can be configured to show video sources from a third-party external video switch in the normal Share Tray view.

The sources will appear—and behave—as any other video source connected directly to the codec. For the user this will be perceived as completely transparent— no video switch seems to be involved.

The video switch feature requires, similar to in-room controls, a third-party control system. The control system will use the Codec API to synchronize the source states between the video switch and the Touch10 user interface using a set of API events and commands.

In order to make this work when the user selects a video source from the Touch10, the codec must be set to issue a corresponding event, which in turn shall cause the controller to send appropriate commands to the video switch and the codec.

This event will be issued only if the controller has registered to the codec upon connection, requesting the following from the codec:

```
xFeedback register Event/UserInterface/Presentation/
   ExternalSource
```

The event issued will be as follows:

```
*e UserInterface Presentation ExternalSource Selected
   SourceIdentifier: "XXXX"
```

Where "XXXX" is a unique string ID used to identify this source when selecting or setting state—see the following pages for more on this.

Furthermore, there are six commands available to control the system:

Add: Adds video source identifiers, including ID of connector, the name to appear on the Touch10, a unique string ID to identify a source when selecting or setting state, and what type of icon to display on Touch10 for each source.

List: Returns the current list of external sources.

Remove: Removes a source from the list.

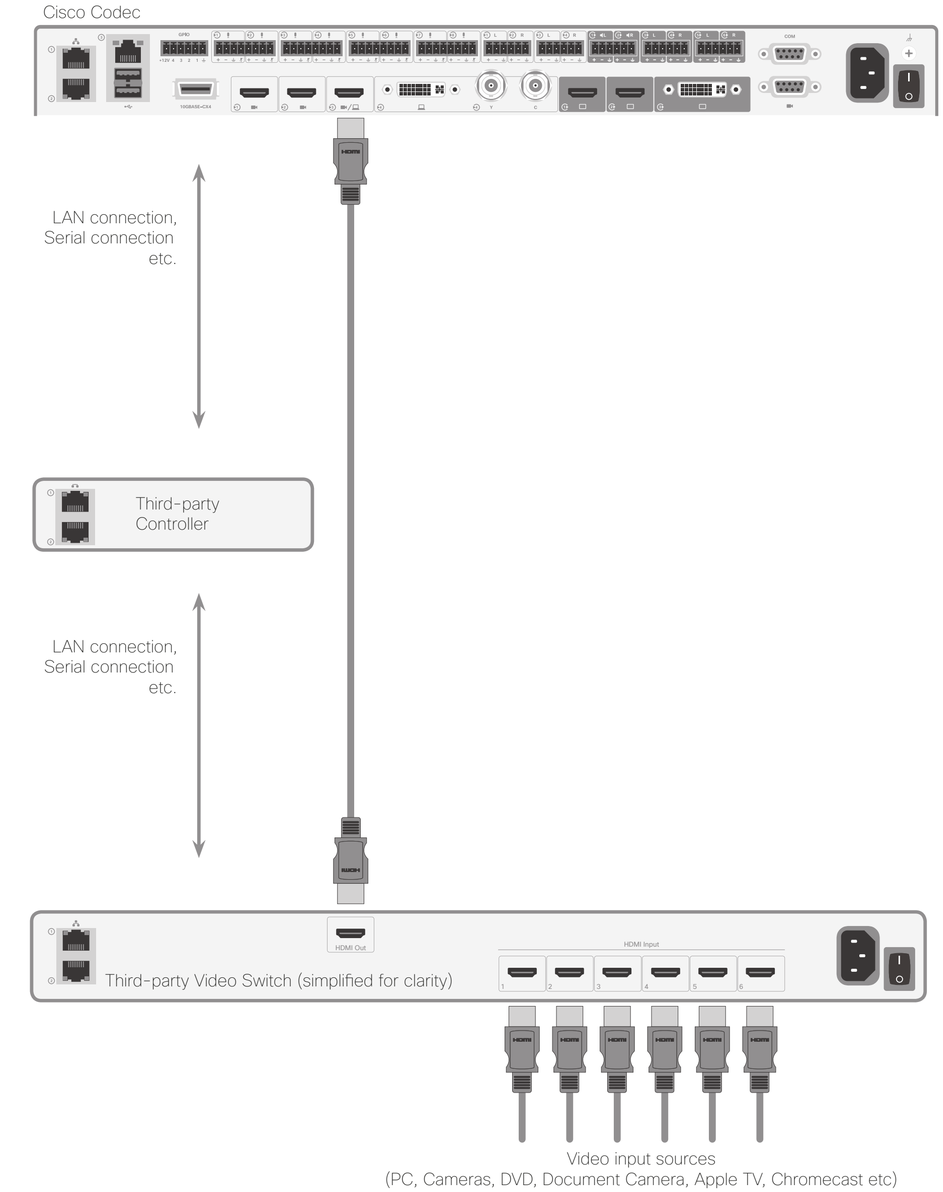
RemoveAll: Removes all of the sources from the list.

Select: Selects a specific source.

State Set: Changes the state of a source.

These are all presented in detail on the following pages.

A simple example of a setup using the configuration shown above is provided in the article "Video Switch Example" on page 46.



Command Details

UserInterface Presentation ExternalSource Add

This command establishes and defines an input source.

```
xcommand UserInterface Presentation ExternalSource Add
   ConnectorId: ConnectorId Name: Name SourceIdentifier:
   SourceIdentifier Type: Type
```

in which:

ConnectorId: The ID of the codec connector to which the external switch is connected

Name: Name displayed on touch 10

SourceIdentifier: A unique string ID used to identify this source when selecting or setting state

Type: Decides what icon is displayed on the Touch 10, choose between: <pc/camera/desktop/document_camera/mediaplayer/other/whiteboard>

Example:

```
xcommand UserInterface Presentation ExternalSource Add
   ConnectorId: 3 Name: “Blu-ray”
   SourceIdentifier: bluray Type: mediaplayer
```

UserInterface Presentation ExternalSource List

This command returns the current list of external sources.

```
xcommand UserInterface Presentation ExternalSource List
```

UserInterface Presentation ExternalSource Remove

This command removes a source from the list.

```
xcommand UserInterface Presentation ExternalSource Remove
   SourceIdentifier: SourceIdentifier
```

in which:

SourceIdentifier is a unique string ID used to identify this source when selecting or setting state.

UserInterface Presentation ExternalSource RemoveAll

This command removes all sources from the list.

```
xcommand UserInterface Presentation ExternalSource RemoveAl
```

UserInterface Presentation ExternalSource Select

Starts to present the selected source if it is in ready state and has a valid ConnectorId. Also shows the item in sharetray as “Presenting”.

```
xcommand UserInterface Presentation ExternalSource Select
   SourceIdentifier: SourceIdentifier
```

in which:

SourceIdentifier is a unique string ID used to identify this source when selecting or setting state.

UserInterface Presentation ExternalSource State Set

Used to change state of the source with SourceIdentifier.

```
xcommand UserInterface Presentation ExternalSource State
   Set SourceIdentifier: SourceIdentifier State: State
   [ErrorReason: ErrorReason]
```

in which:

SourceIdentifier: is a unique string ID used to identify this source when selecting or setting state

State: <Error/Hidden/NotReady/Ready> Ready is the only presentable state, hidden exists in the list but does not show in the sharetray.

ErrorReason: Optional. Displays in the share tray if the state is set to Error.

Video Switch Example

A simple example of setup could be:

Controller sending:

```
xcommand UserInterface Presentation ExternalSource Add
   ConnectorId: 3 Name: “Blu-ray” SourceIdentifier: bluray Type:
   mediaplayer

xcommand UserInterface Presentation ExternalSource Add
   ConnectorId: 3 Name: “Apple TV” SourceIdentifier: appletv Type:
   mediaplayer

xcommand UserInterface Presentation ExternalSource Add
   ConnectorId: 3 Name: “TV” SourceIdentifier: tv Type:
   mediaplayer
```

The default state is NotReady (Fig. 1)

So the next step for an integrator would be to set them to ready (Fig. 2).

```
xcommand UserInterface Presentation ExternalSource State Set
   State: Ready SourceIdentifier: bluray

xcommand UserInterface Presentation ExternalSource State Set
   State: Ready SourceIdentifier: appletv

xcommand UserInterface Presentation ExternalSource State Set
   State: Ready SourceIdentifier: tv
```

If one of the sources is selected on the video switch the controller should send a command accordingly:

```
xcommand UserInterface Presentation ExternalSource Select
   SourceIdentifier: tv
```

If the switch is connected on the chosen connector it will start to present (Fig. 3).

When a user selects another source, by clicking the other source item in the share tray, the codec will send the following event:

```
*e UserInterface Presentation ExternalSource Selected
   SourceIdentifier: “appletv”
```

The Controller should listen to this event and display the selected source.

Note! UPDATED The presentation will not start if the below setting has been set to Manual:

```
xconfiguration Video Input Connector [x]
   PresentationSelection: <AutoShare, Desktop, Manual, OnConnect>
```

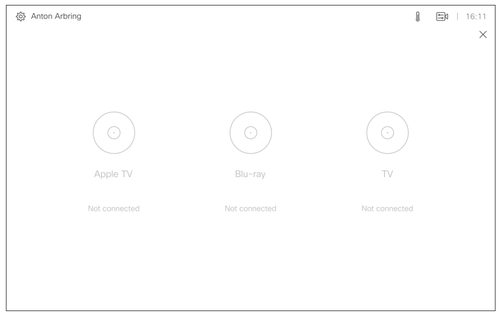


Fig. 1 Default state is NotReady.

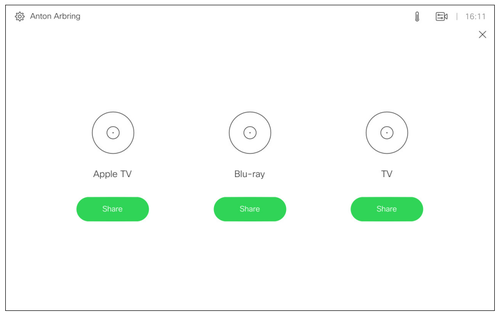


Fig. 2 When Input sources have been set to Ready.

Fig. 3 If the switch is connected on the chosen connector it will start to present.

Part 4
Working with Macros

Creating Macros

Search
Introduction
UI Extensions
API
Widgets
CMD Reference
Messages
Trouble-shooting
Tips & Tricks
Room Simulator
Video Switch
Macros

Macros allow you to write snippets of JavaScript code that can automate parts of your video endpoint, thus creating custom behavior.

Kindly observe the following:

- The SX10 does not support macros.
- Macros do not run on Cisco Webex (formerly Cisco Spark) enabled systems.

Sign in to the video system's web interface with administrator credentials and navigate to **Integration** > **Macro Editor**.

The first time this is done on a codec, you will be asked whether to enable the use of macros on this codec.

You may later disable the use of macros from within the Macro Editor. This will not permanently disable macros from running. Every time the codec is restarted, the macros will be re-enabled automatically.

To disable automatic restart, you must use the `xConfiguration Macros Mode: Off`.

You may want to use this command in the event of unintended behavior by the system. In such cases you should always disable the macros before proceeding with your troubleshooting.

The Macro Framework has many benefits, as it allows integrators to:

- Tailor their deployments.
- Create their own "features" or "workarounds" to functionality Cisco is reluctant to provide in form of a new software feature.
- Automate scenarios / re-configurations.
- Create custom tests or monitoring.

With the Macros, in-room controls no longer needs an external control system to activate local *functionality*.

However, performing local *actions* via the xAPI, such as to control other things like lights and blinds will still require a suitable third-party control system.

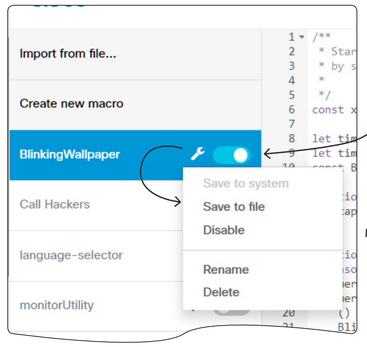
Examples of local *functionality* can be an In-Room Control panel for speed dialing or to trigger a "Room Reset" that puts all the configurations back to default (input sources, camera presets etc.).

Note: The Macro Framework is limited to local xAPI interaction. You cannot establish remote network connections to servers to push or receive data via the Macro Framework code.

Disclaimer: Cisco will support the Macro Framework itself only. Cisco will not support code that fails to compile or fails to work as the developer "intended". It is entirely up to the person writing the code to ensure that the syntax is correct and that possessed coding skills are sufficient to write macros in JavaScript.

Please refer to public developer forums, the API Reference Guide of the product in question, as well as the Help section of the Macro editor. See also the following pages for more.

The Macro Editor Panel



Activate or deactivate a macro.

Above shows where to start a new macro programming session from. You can either import existing code from a file (***.js**), by clicking **Import from file...** or create a blank macro by clicking **Create new macro**.

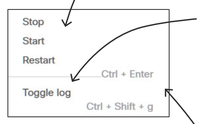
By clicking **Create new macro**, the main editor window will be activated. You may now start coding.

The macro will be displayed in the list of macros.

Click the **Wrench** icon to gain access to available options, as shown above.

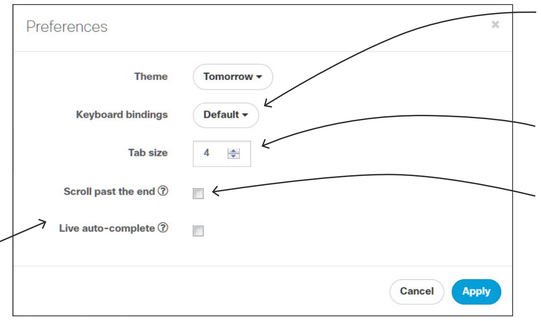
To change the name of a macro, you may also click its name, edit it, and then hit **Enter**.

Anytime a macro is saved, deleted or activated/ deactivated, the whole runtime (and therefore all active macros) is restarted. Read more about runtime on the following page.



Use these to start, stop or restart the macros. Note that after a restart, macros will be re-enabled automatically, see text on the previous page for more on this.

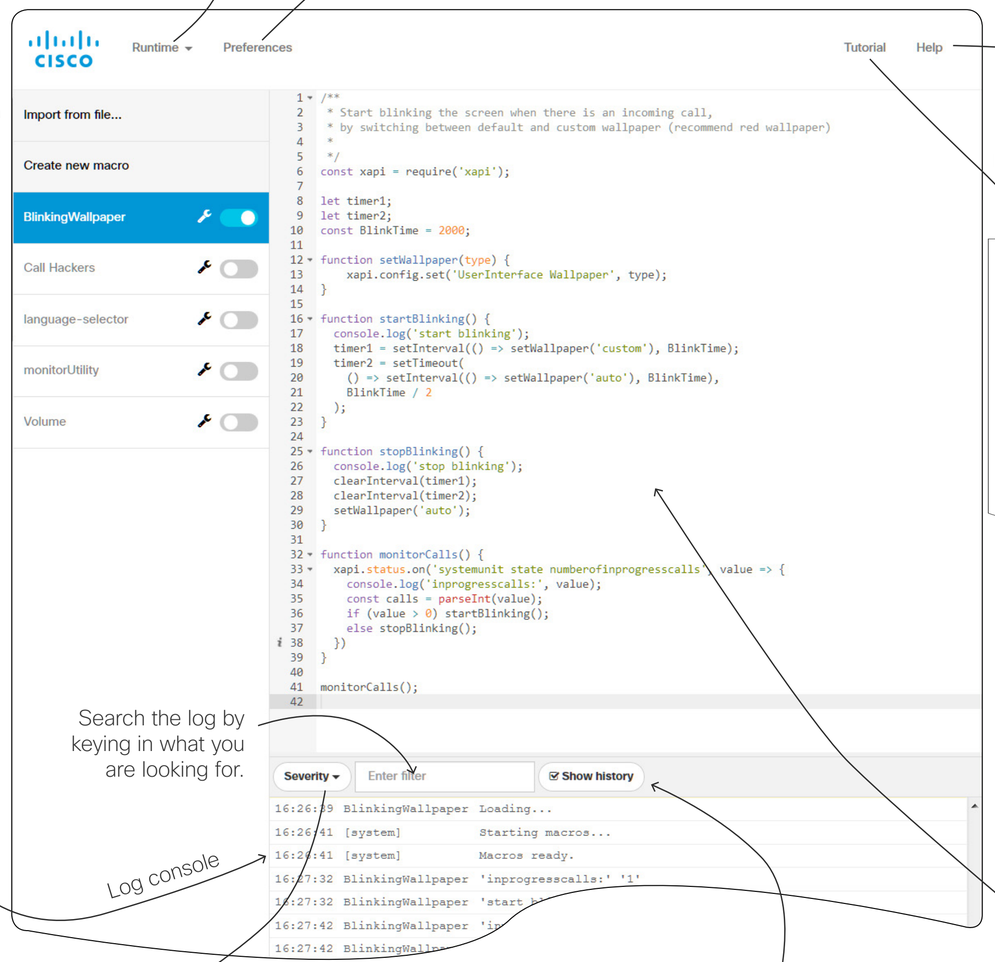
Toggle log will hide / display the Log Console.



Defines which set of keyboard shortcuts to use, choose between **Vim** or **Emacs**.

Defines the width of a tab (in # spaces).

When checked, **Scroll past the end** keeps the bottom lines of your macro in the middle of your screen.



Search the log by keying in what you are looking for.

Log console

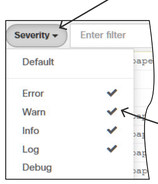
Check **Show history** if you want to the log to cover the entire history and not just what has taken place since the last restart.

The purpose of the **Log Console** is to reveal what happens when you run the macro. Here you will see the actions of the runtime and whatever you choose to print out to the console.

Much of what is displayed in the console log window is exported to the **macros.log** file in the log bundle of the endpoint.

This can be used to reveal errors and exceptions in the code. You can also log custom text by issuing:

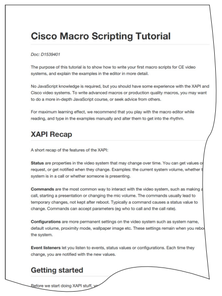
```
console.log('this is a log entry');
```



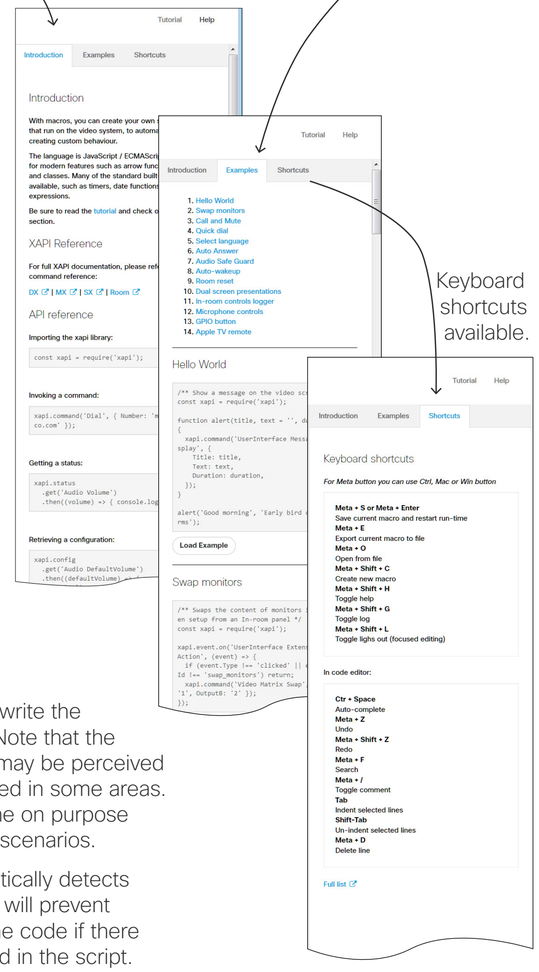
Make sure the type of errors and exceptions you want to be logged is checked here, otherwise they won't appear in the console log window.

Macros tutorial

Introduction to macros



Clicking on the Tutorial will prompt you to download a PDF.



This is where you write the JavaScript code. Note that the JavaScript library may be perceived as somewhat limited in some areas. This has been done on purpose to prevent certain scenarios.

The editor automatically detects syntax errors, and will prevent you from saving the code if there are errors detected in the script. Hoover over the error to see details.

For any of the example macros you can click the **Load Example** in order to paste the code directly into the main editor window.

Note: You must be in an active editing session for this to work, click **Create new macro** from the left menu and then click **Load Example**.

Things to Observe

Macros can alter the expected behavior of the system. If your macro is likely to surprise or confuse normal users, let them know with e.g. a notification on the video system screen.

We recommend that you never let macros depend on other macros. You can, of course, create several macros that listen to the same xapi values, e.g. the call state, as long as their actions are independent of each other.

Note that extensive use of macros may slow down codec performance due to heavy load.

Currently, macros cannot get or send data externally, e.g. to control lights in the room. For this you will need an external control system. However, it can be useful to combine macros and external systems, for example using a Crestron for light control at a low level, and then use macros to adjust in a more sophisticated way, such as adjusting the light depending on presentation status, call status etc.

Your macros may contain customized text to be displayed on the Touch10/DX. This text may alert users to observe the activation or deactivation of certain features as well as alert them to act in accordance with the message given, etc.

Such text can be purely informative, but it may also prompt the user to respond to it by keying in information. This information may, in turn, cause the video system to directly act upon it. Note that since macros cannot get or send data externally, this feature cannot be used to send relevant information outside the codec, e.g. to a predefined URL.

About Macro Runtime

All the activated macros run in a single process on the video endpoint, called the macro runtime. It should be running by default, but you can choose to stop and start it manually from the editor. If you restart the video endpoint, the runtime will automatically start again if `xconfiguration macros autostart` is **On**.

If any macro becomes unresponsive (fails to respond within a few seconds due to e.g. an infinite loop), a safety mechanism will stop the runtime, thereby stopping all macros.

The runtime will then be automatically restarted after a few seconds. This will continue, but the time between restarts will increase every time the runtime is shut down. If this happens more than a certain number of times, this will cause a system diagnostic to be displayed to notify that the macros are having problems.

Learning Resources Available

If you want to learn about how to utilize the macros feature, may we recommend the following:

- Read through the *Introduction to Macros*, which can be found in the Help section of the Macro Editor.
- Read through the *Macros Tutorial*, which is also found in the Macro Editor. This tutorial is also available as a free download from cisco.com.
- The Macro Editor even contains several examples ready for use. All these examples may be loaded in to the editor and studied there, or they may be used as is in your configurations. They may, of course, also serve as basis for further refinement, if you so wish.

Disable Macros When Troubleshooting

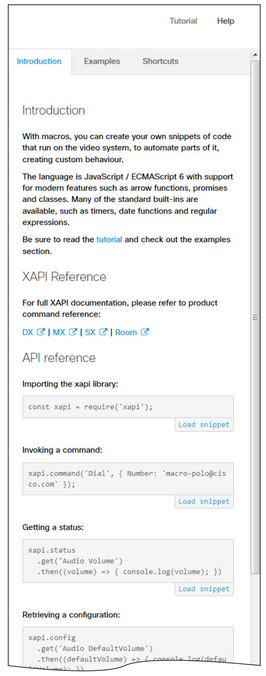
If you experience unintended behavioral changes and you run macros on your system, make sure you deactivate the macros before proceeding with your troubleshooting.

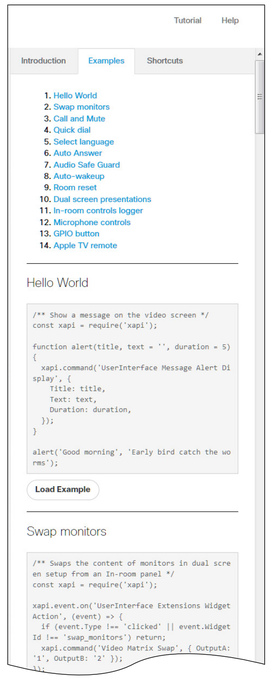
Use `xConfiguration Macros Mode: On/Off` to do this.

The macro framework has its own log file called **macros.log**

The **macros.log** file contains much of what is printed in the Macro console. The macros can be configured to print output to the console and this will be stored in the log, so keep in mind that you can see custom log messages (which must have been created by the developer) in this file.

An *Introduction to Macros* can be found in the Help section of Macro Editor.

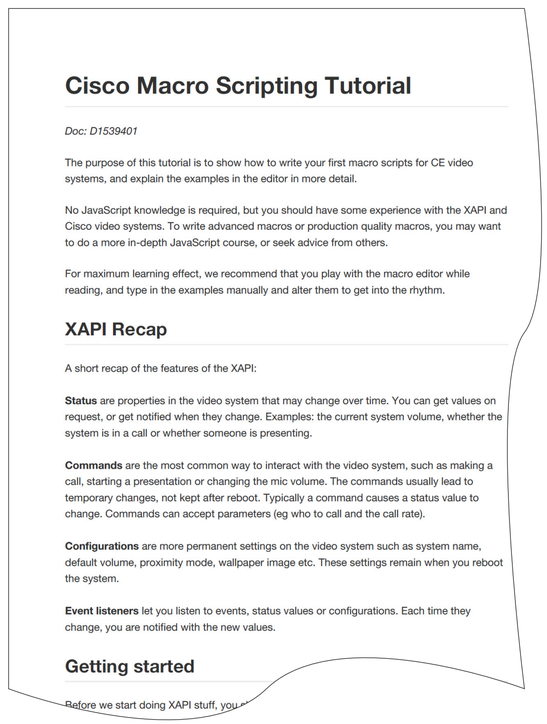




For any of the example macros you can click the **Load Example** in order to paste the code directly into the main editor window. See the previous page for more.

The *Tutorial* provides most of what you will need to know to start creating your own macros.

Note that clicking on the Tutorial will prompt you to download a PDF.



Cisco contacts

On our web site you will find an overview of the worldwide Cisco contacts.

Go to: http://www.cisco.com/go/offices

Corporate Headquarters
Cisco Systems, Inc.
170 West Tasman Dr.
San Jose, CA 95134 USA

Intellectual property rights

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

All printed copies and duplicate soft copies are considered un-Controlled copies and the original on-line version should be referred to for latest version.

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco website at www.cisco.com/go/offices.