In-Room Control on Touch 10
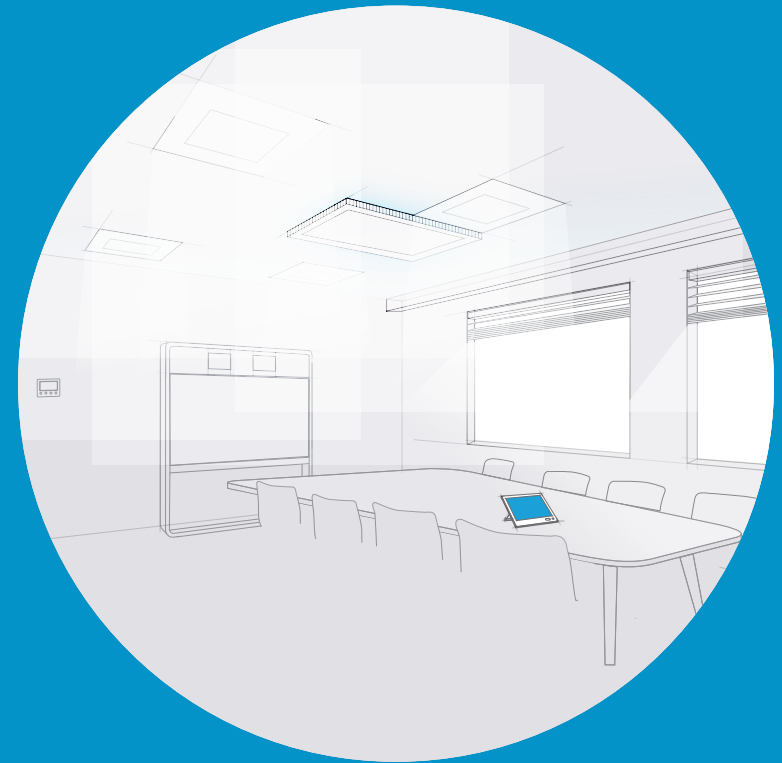
User Interface Extensions

Cisco TelePresence

MX200 G2, MX300 G2, MX700, MX800, SX10, SX20, SX80

CE8.3

December 2016

CISCO

Table of contents

How to use this guide

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 and 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.

Introduction

Introduction

With In-Room Control you can add custom elements to our Touch 10 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 Touch 10 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 SX Series and MX Series video systems running Collaboration Endpoint Software, version CE8.2 or later.

Definition of terms

Video system. Video system or codec in the Cisco TelePresence MX Series or SX Series running Collaboration Endpoint Software, version CE8.2 or later. Sometimes referred to as *video device*.

Control system. Third-party control system with hardware drivers for peripherals, for example Crestron, AMX, Apple HomeKit, or Android.

Touch 10. Out touch-based control device for the MX Series and SX Series video systems. Full product name: Cisco TelePresence Touch 10. Also known as Touch 10 controller, or Touch 10 user interface.

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 Touch 10. 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 Touch 10.

In-room controls

Global entry icon example

Global control panel example

Homescreen entry icon example

Example of Global and Homescreen in-room control panels on Touch 10. Any In-Call panel icons existing will be visible during calls only.

You can customize the Touch 10 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.

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

· **Global** panel has its entry icon in the status bar at the top of the Touch 10 display. Once established, this entry icon is visible at all times.
· **Homescreen** panel has its entry icon located to the right of the buttons appearing along the bottom of the Touch 10 display. This entry icon is visible outside calls only.
· **In-Call** panel has its entry icon located to the right of the buttons appearing along the bottom of the Touch 10 display. This entry icon is visible when in a call only.

You need a Cisco *video system* with a *Touch 10* 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 *in-room control 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 Touch 10 user interface.

Together, all of this provides a powerful combination of the control system's functionality and the user-friendly Touch 10 user interface.
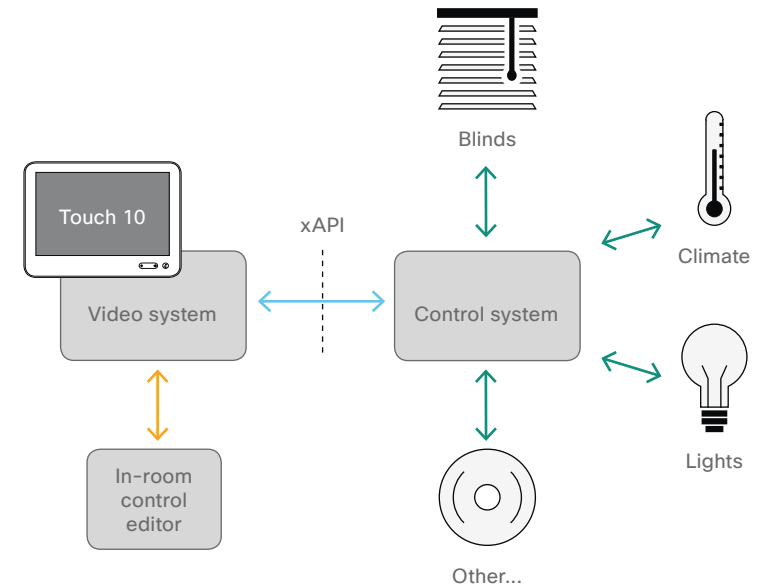
In-room control schematics

Create a user interface

Use the in-room control editor to create the user interface and to apply it to
the video system

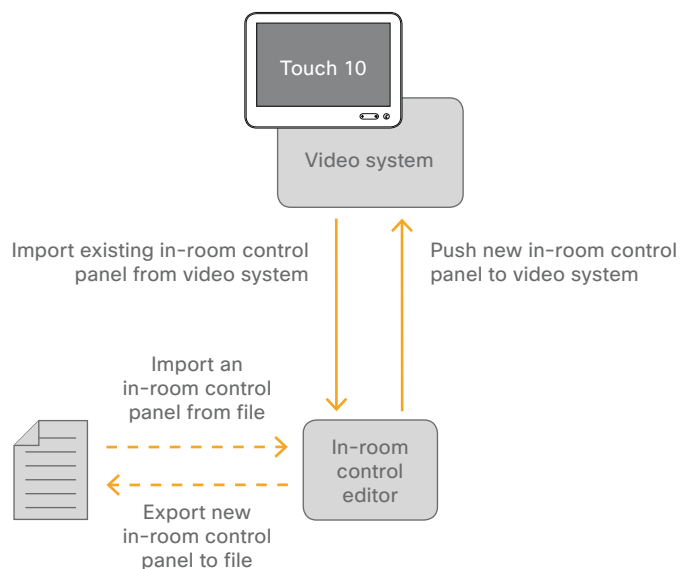**CISCO**

Create a user interface for the Touch 10

Use the *in-room control editor* to create customized panels for peripheral controls on the video system's Touch 10 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 Touch 10, 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 Touch 10.

Touch 10

Video system

Import existing in-room control panel from video system

Push new in-room control panel to video system

Import an in-room control panel from file

In-room control editor

Export new in-room control panel to file

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 Touch 10. Apart from this, the offline editor has full functionality.

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.

Import an in-room control panel from file

Offline in-room control editor

Export new in-room control panel to file
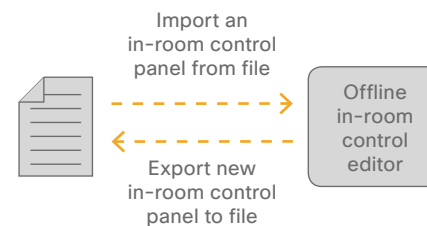
Offline in-room control editor

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, open the *index.html* file that you find in the *rceditor* folder.

Import the current panel from the video system (codec), or import a panel that has been exported to a file on an earlier occasion →

Export your panel to the video system (codec), or export it to a file that can be imported back into the editor at a later stage →

Link to online guides →

📑 Panels

⬇ Import

⬆ Export

? Help

Configure your panels

With the In-Room Control editor, you can create custom panels on your Cisco Touch 10. You can put your features in 3 different places, depending on what they do:

Global

The features you put here will show up in the system bar. They will always be available, both when you're in a call and when you're not.

Recommended features:

💡 Lights

⚙ Blinds

🌡 Ventilation

⊕ Create

Homescreen

These features will only be available outside call on homescreen. Anything media or call initiation related might be placed here.

Recommended features:

🖵 TV

🍽 Services

♪ Media

⊕ Create

In-Call

Only features that are needed in call and those that affect the video conference should be placed here.

Recommended features:

🎤 Microphones

📼 Recording

? Help

⊕ Create

There are three sets of panels available; Global, Homescreen and In-Call. Each set may contain multiple pages. To create a panel, click here.
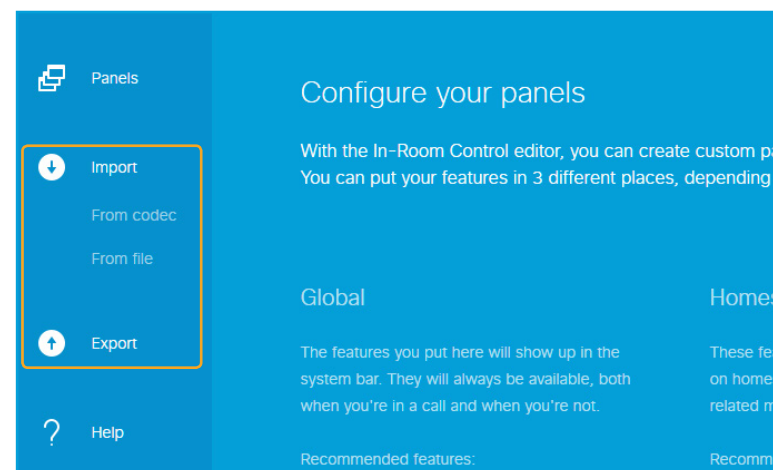
cisco

Create an In-room Control panel

Start Designing the In-room Control Panel

If an existing in-room control panel failed to load automatically on launching the editor, choose one of these alternatives:

- *Create* creates a blank in-room control panel in the editor for either Global use, Homescreen use or In-Call use.
- *Import > From file* loads an in-room control panel, which has been exported to a local file on an earlier occasion, into the editor.
- *Import > From codec* loads the current in-room control panel from the video system into the editor. Not available in 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.

Start or continue the design of an in-room control panel

Populating the In-room Control panel

What you see in the editor's design panel is similar to how the in-room control panel will appear on the Touch 10.

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

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 fits in one line, widgets wrap to a new line within the same row.

How you use the user interface elements for your meeting room is up to you. You can, for example, create panels with preset buttons for lights and blinds, or support for more complex scenarios such as controls for many microphones in a large meeting room. You can find some examples for inspiration in the Examples chapter.

More on how to create the panel can be found on the next page.

This is how a panel looks before you have started to populate it.

Any panels already created, appears in a list beneath the Panels button. To create more panels, click *Panels*. This will show the panel front page again. Choose a new panel to create, or click an existing one to continue editing it.

Buttons with familiar symbols like home, power, arrows, camera and loudspeaker controls, media player controls

Buttons with custom text, group buttons, toggle button, sliders, spinners, and other elements

Panels
Global

Import

Export

? Help

Global

Panel icon

General   Icons

On / off   Slider

Up / down

Short   Medium

Wide

2 Options   3 Options

4 Options

21°C   Aa
Value   Short

Aa   Aa
Medium   Wide

Page name

Enter row name

+

Four-column grid

Design panel

Widget library

Page shows a setup of a Global in-room control, but the principles apply to all three sets of in-room controls.

1 Give the page a name and ID

Click the page text, and enter a name. You should also give the page a unique Page ID, which can be used to issue an event when the page is opened or closed. The Page ID field is only shown when you select the Page name field (not its text) and will then appear where the Widget ID is shown in the below Fig.

2 Give the row a name

Click the row text, and enter a name.

3 Add widgets from the library

Drag and drop as many widgets as you need from the widget library onto a row.

4 Enter custom text on widgets

Click the dummy-text on a button, group button, or text field, and enter the appropriate text.

5 Define unique identifiers

Click the widget, and type unique identifiers in the input fields that pops up below the design panel. You can either use the pre-assigned identifiers or define your own.

All widgets have a unique *Widget ID.* Only Group buttons have *Group IDs.*

Read more about the identifiers in the Widgets chapter.

6 Add more rows

Click ⊕ to add a new row.

Repeat steps 2-5 for each row.

7 Add more pages

Click ⊕ to add a new page.

Repeat steps 1–6 for each page.

8 Choose an entry icon

Click *Panel icon* and choose a control icon (entry icon).

If you created a Global panel, the control icon appears in the status bar on Touch 10. When you tap the icon, the in-room Global control panel opens.

For Homescreen and In-Call, the entry icon will appear as a button to the right of the call control buttons.
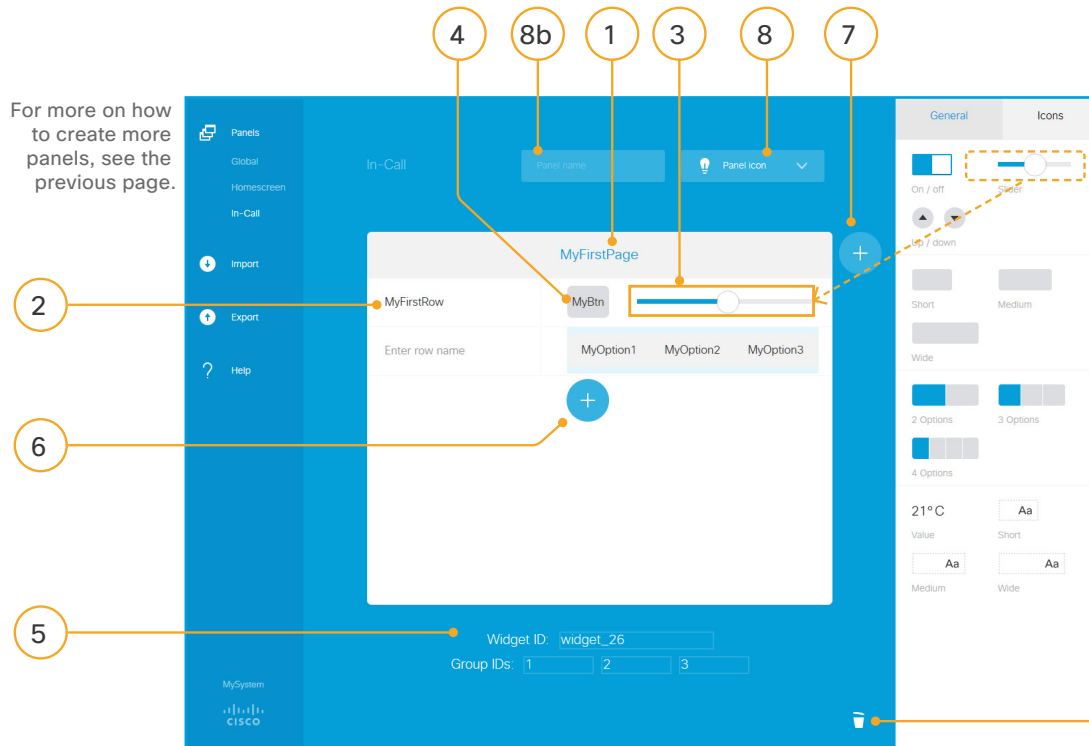
8b Give the panel a name

Click *Panel name* and type in a descriptive name. This field is not available with the Global panel.

9 **Click on the garbage bin icon** to delete the entire panel from the editor. To remove a panel from the Touch 10, remove it from the editor and then push your altered editor configuration to the video system. The panel's icon should disappear instantly.

For more on how to create more panels, see the previous page.

| Panels | Global | Homescreen | In-Call |

In-Call

Panel name | Panel icon ⌄

General | Icons

On / off | Slider
Up / down
Short | Medium
Wide
2 Options | 3 Options
4 Options
21°C | Aa
Value | Short
Aa | Aa
Medium | Wide

MyFirstPage

MyFirstRow | MyBtn

Enter row name | MyOption1 | MyOption2 | MyOption3

Import
Export
Help

MySystem

Widget ID: widget_26
Group IDs: 1 | 2 | 3

Tip Drag a page, row, or widget, and drop it in a new position. Drop it outside the design panel to delete it.

CISCO

Apply In-room Control panel to the video system

(Not available when using the offline editor. Instead, you should export the in-room control panel to a file in order to save your work for later.)
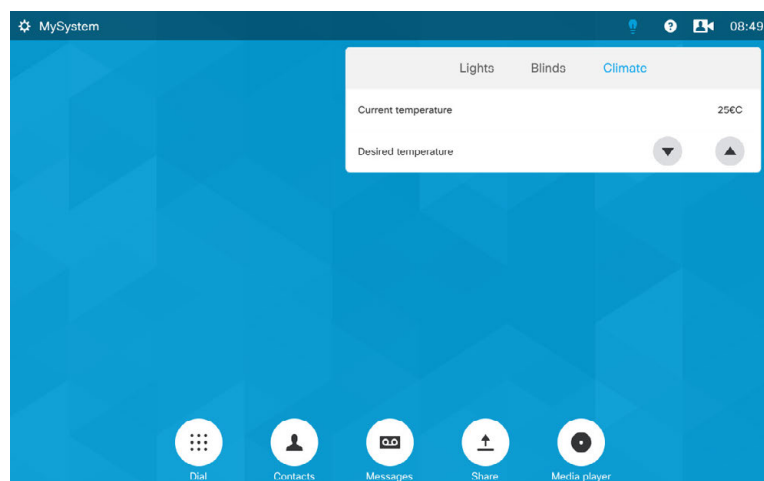
Apply the in-room control panel to the video system. You will see the results immediately on the Touch 10.

· Choose *Export > To codec*

Check the result on Touch 10

(Not available when using the offline editor.)

Tap the in-room control icon in the Touch 10 status bar to open the in-room control panel.

Example of two sets of in-room control panels. The Global has been selected (blue light bulb) and a panel consisting of three panes is shown.

To the right of the Share button, a in-room control Homescreen entry icon has been added.

Which in-room control icons (entry icons) used is defined by the *Panel icon* setting, as shown on the previous page.

Export the In-room Control panel to file

You may also export the in-room control panel to a file. This file can be imported back into the editor at a later stage. Note that you will export all your work (Global, Homescreen and In-Call panels)

· Choose *Export > To file*

If you are using the offline editor, this is the only way you can save your work for later.

Exporting to file may also be useful as a way of creating a design template if you are going to push the same in-room control panel to multiple video systems.

Files exported while connected to a video system are in the same format as files exported from an offline editor. This means that any of these files can be imported into an editor launched from a video system's web interface, and then pushed to the video system's Touch 10.

Remove a panel from Touch 10

To remove a panel from the Touch 10, remove it from the editor and then push your editor configuration to the video system. The panel's icon should disappear instantly.

API for in-room control

CISCO

13

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
- Ethernet port
- 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, or

http://www.cisco.com/go/mx-docs for MX 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 Touch 10 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 Touch 10 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.
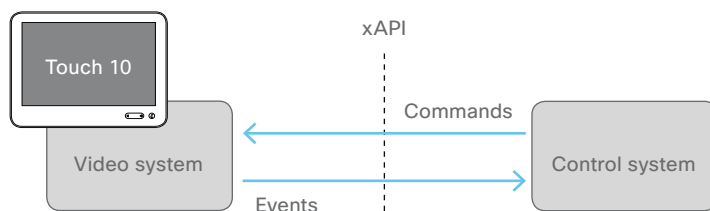
The video system and the control system exchange messages through the xAPI.

Examples:

· When someone taps a *Lights On* button on Touch 10, 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 switch 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 Touch 10 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 the video system and the control system

You can register the control system as a peripheral that is 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.

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.

Observe that 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.

CISCO

15

Events for widget actions

The video system sends one or more of the following events when someone uses the controls on the Touch 10 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 below 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.
 	
 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```
 ↑ *Type of action*
	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/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>```
	 ↑ *Type of action*

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

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 Touch 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 above example. 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.

CISCO

17

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 Touch 10 in-room control panel are updated accordingly.

It is important that the control system sends `SetValue` commands in the following situations, so that the Touch 10 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 Touch 10 in-room control panel.
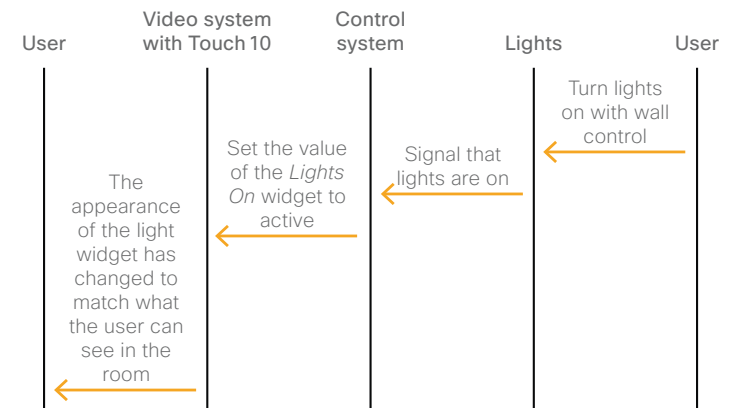
   The control system must also do all that is necessary in the room to reflect the action on the Touch 10 in-room control panel, for example switch on the light.

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

Examples
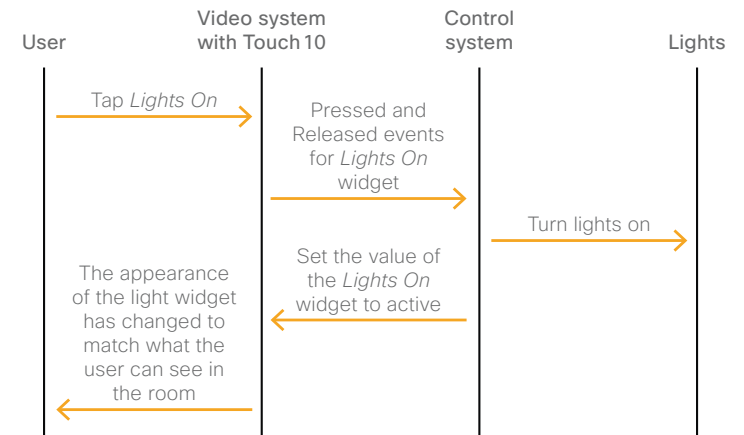
Message flow—turn on the lights using the controls on Touch 10

Message flow—turn on the lights using the wall control

Widgets

Description of the user interface elements and their associated events and commands

Widgets

The Touch 10 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.

The content of the next few pages describes each widget type, 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 Touch 10 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 40.

The Widget ID is the programming link between Touch 10, 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 ID*s are unique identifiers for the individual buttons within the group.

Widget_ID_groupbutton		
Option 1	Option 2	Option 3
Group_ID_one	Group_ID_two	Group_ID_three
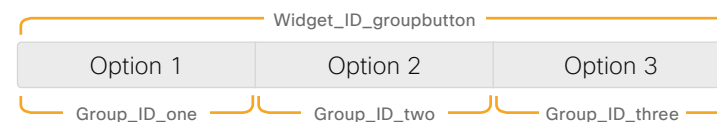
A Group ID is assigned automatically, but can be defined by you instead. A Group ID can be any name or number; we recommend using a descriptive name without special characters. The maximum number of characters is 255.

Toggle button

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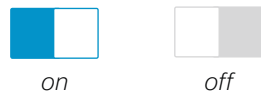
A two-state switch which indicates either *on* or *off*.

Example of use: Anything that can be turned on or off, for example lights, fan, and projector. You can use a toggle button together with a slider for lights that can be dimmed.

Events

Changed   Triggered when the button is released.
           Value: <on/off>

Example: Press "on" on the toggle button 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 the button with WidgetId = "togglebutton" to "on".

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

CISCO

Slider

Example of use: Dimmable lights, volume control.

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.

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"
```

Spinner

▲ [Aa...] ▼

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

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.

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"
```

23

Button with custom text

Text	Text
inactive	*active*

Example of use: Room presets for lighting, for example three buttons that show *Dark, Cool, Bright*.

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.

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 = "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"
```

Button with symbol

inactive   active

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

There are buttons with different symbols under the *Icons* tab in the library. Such a button has similar behavior as a button with 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.

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"
```

25

Group button

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.

| Text | Text | Text |
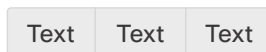
No button selected

| Text | Text | Text |

One button selected

Groups of either two, three or four buttons with custom text are available. The buttons within a group are linked so that only one can be selected at a time (also known as radio buttons).

The size of the button 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.

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"
```

CISCO

Text box

Aa   Aa

21°C

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.

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.

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."
```

CISCO

Command reference

Syntax and semantics for events, commands, and statuses, that are used with the user interface extensions

28

Command reference

This chapter includes syntax and semantics for events, commands, and statuses, that are used with the user interface extensions.

Refer to the API guide to find descriptions of all configurations, commands, and statuses. Go to

> http://www.cisco.com/go/sx-docs for SX Series,

> http://www.cisco.com/go/mx-docs for MX Series

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

Overview of events, commands and statuses

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`*

where

`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`*

where

`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`*

where

`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`*

where

`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 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 it equivalent to the UserInterface Extensions Event Pressed, UserInterface Extensions Event Changed, UserInterface Extensions Event Released, or 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>
```

where

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
```

where

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
```

where

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
```

Use of Video Switch

34

Using a 3rd Party Video Switch to Extend the No. of Video Sources Available

The Cisco Touch 10 panel can now be configured to show video sources from a 3rd 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 users it will be transparent that there is a video switch involved.
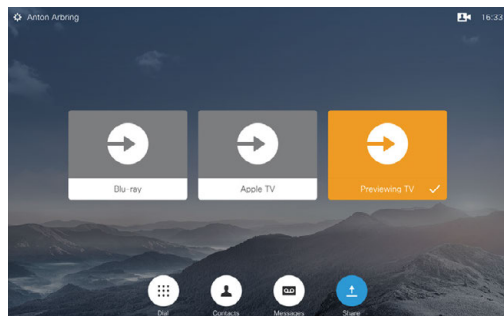
The video switch feature requires, similar to in-room controls, a third party control system. The control system will use the Codec API to syncronize the source states between the video switch and the touch 10 user interface using a set of API events and commands.

In order to make this work when the user selects a video source from the Touch 10, 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.

Cisco Codec

LAN connection, Serial connection etc.

Third-party Controller

LAN connection, Serial connection etc.

HDMI Out   HDMI Input

Third-party Video Switch (simplified for clarity)   1 2 3 4 5 6

Video input sources
(PC, Cameras, DVD, Document Camera, Apple TV, Chromecast etc)
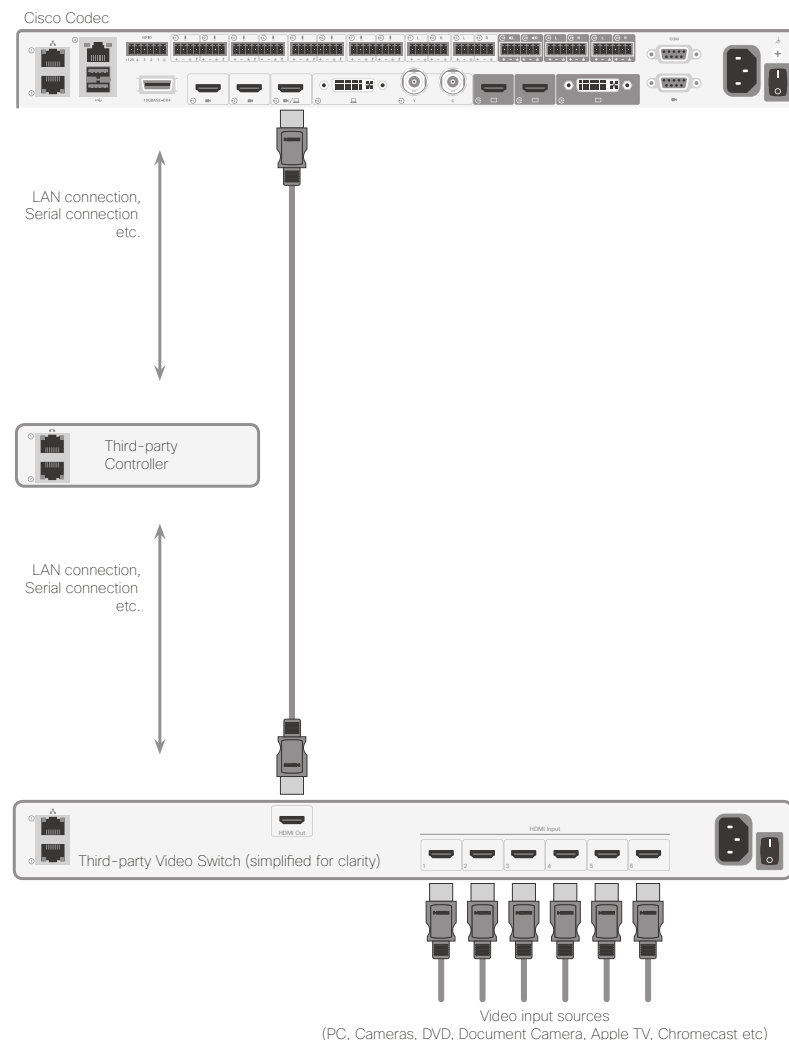
Furthermore, there are six commands available to control the system:

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 49.

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
```

where:

   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 Touch10, 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
```
where
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 RemoveAll
```

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
```

where

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]
```

where:

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.

Troubleshooting

Help to identify the source of an error

Troubleshooting

Development tools on the web interface

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 Touch 10 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.

Widget status

Widget State Overview

ID	Type	Current Value	Update Value	
lights_on	Button	inactive		Unset
lights_off	Button	inactive		Unset
spotlights_slider	Slider	18		Unset
spotlights_on_off	ToggleButton	off	off	Unset
			off	
			on	

Send command to update widget

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.

Log   Filter logs by widget id…   Clear log

```
Monitor started…
Fetched In-Room Control configuration…
*e UserInterface Extensions Widget Action WidgetId: spotlights_on_off Type: changed Value: on      [spotlights_on_off]
*e UserInterface Extensions Widget Action WidgetId: spotlights_slider Type: pressed                [spotlights_slider]
*e UserInterface Extensions Widget Action WidgetId: spotlights_slider Type: changed Value: 103     [spotlights_slider]
*e UserInterface Extensions Widget Action WidgetId: spotlights_slider Type: released Value: 103    [spotlights_slider]
*s UserInterface Extensions Widget 15 Value: on                                                    [spotlights_on_off]
*s UserInterface Extensions Widget 16 Value: 8                                                     [spotlights_slider]
*s UserInterface Extensions Widget 16 Value: 9                                                     [spotlights_slider]
*s UserInterface Extensions Widget 16 Value: 10                                                    [spotlights_slider]
*s UserInterface Extensions Widget 16 Value: 11                                                    [spotlights_slider]
```

Events

Status

Widget ID

Tips, Tricks & Examples

For inspiration and guidance on best practices

Tips and tricks

Re-register to feedback from the video system when either the video system or the control system restarts

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 Touch 10 in-room controls or pushes a new in-room control panel to the Touch 10.

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

It is important that the control system initializes all the widgets on the Touch 10 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 Touch 10 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 Touch 10.

For example, it makes no difference if you use a slider on the Touch 10 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.

a. Launch the in-room control editor from the video system's web interface.

b. Create the in-room control panel you want, or import a previously saved panel from file (*Import > From file*).

c. Click *Export > To codec*.

Other useful stuff

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

For blinds, consider to use the following strategy: Short press on a direction arrow tilts the blinds. If tilted all the way already, moves the blinds slightly.

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

CISCO

43

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

Remove an 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 Touch 10 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.

Perform the following steps to remove the in-room control panel and icon from Touch 10:

    a. Launch the in-room control editor from the video system's web interface.

    b. Select the panel to be removed (Global, Homescreen or In-call)

    c. Click *Recycle bin* (⬚)in the lower right corner.

Transitioning 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 touch 10".
- Reprogram the signalling from the button-presses coming from the third party control system panel so that it listens to button presses from the touch 10" 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.

Examples

The following examples are meant for inspiration and to give some guidance on best practices. It is not mandatory to design and implement controls as illustrated in these examples.

Group controls that belong 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.
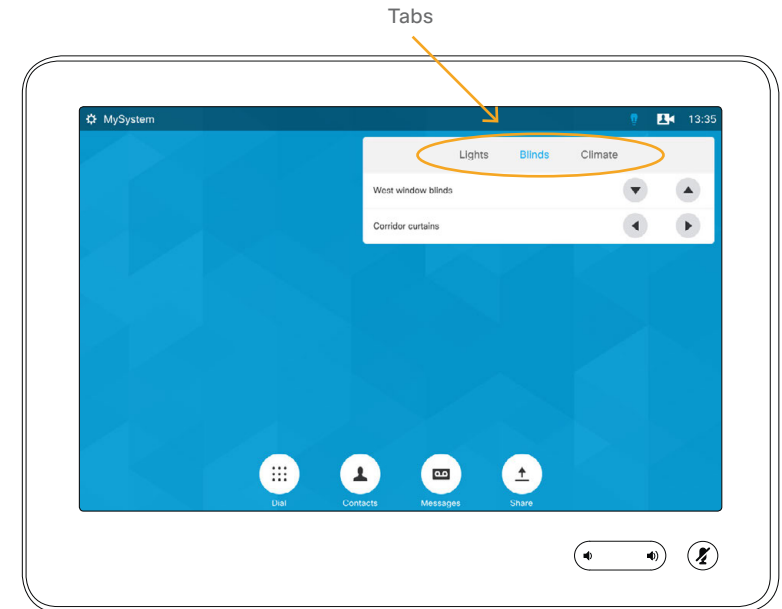
Tabs

45

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.

Combination of slider and toggle button
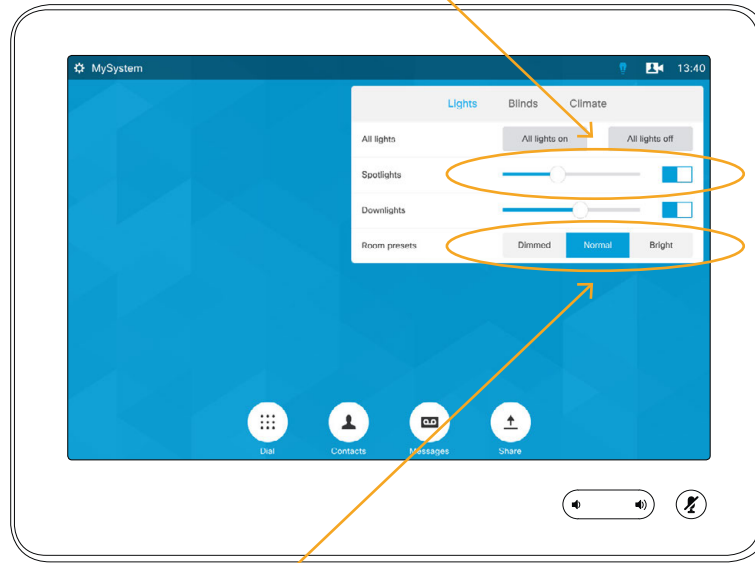
Group button with three mutually exclusive options

Consider the following strategy:

· Set the slider to minimum when the user turns the lights off.
· Set the 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.

Large font text box
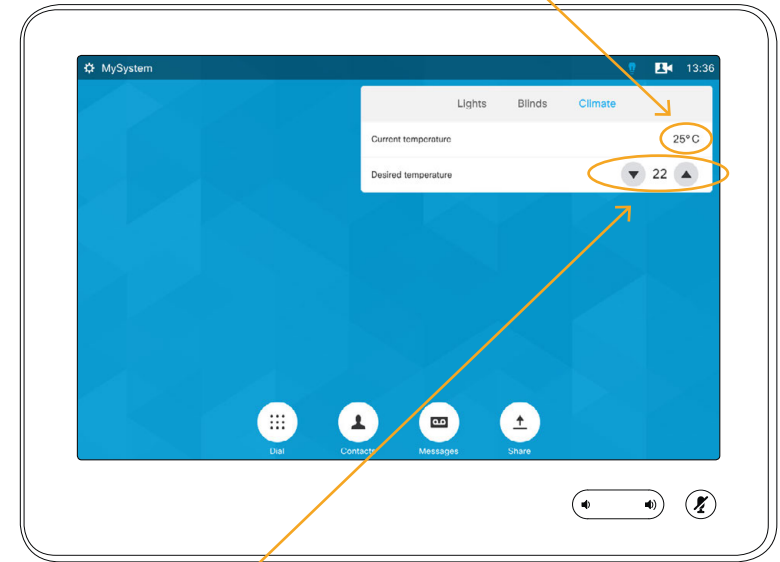
A spinner with text between the up and down arrows

For the best user experience remember the following:

· 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.
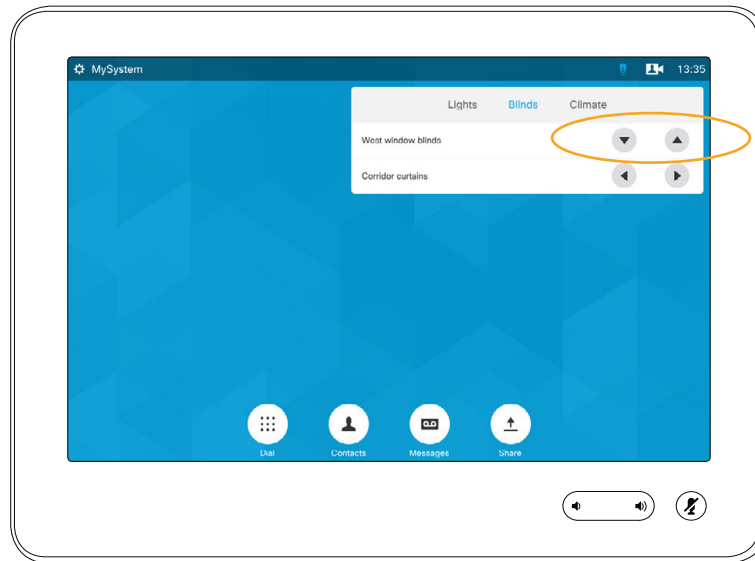
Consider the following strategy:

· 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 is necessary.

Use a button only to send a signal

A button can be used just to send a signal, without changing the button's state or visual appearance.

As an example, use the *All lights on* button to switch on all lights. Do not change the button's color from gray to blue, but update the other light controls to reflect the true status of the room.

A button that does not change color
(from gray to blue) when used
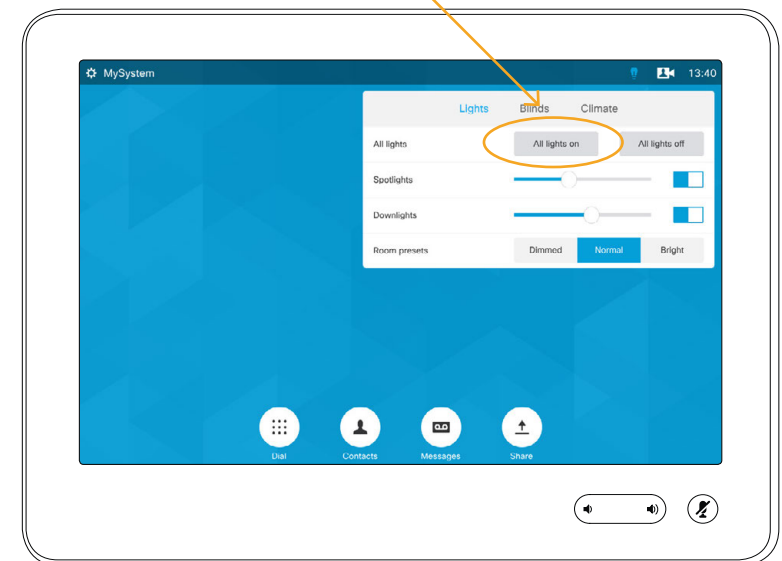
Group buttons

Group 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.
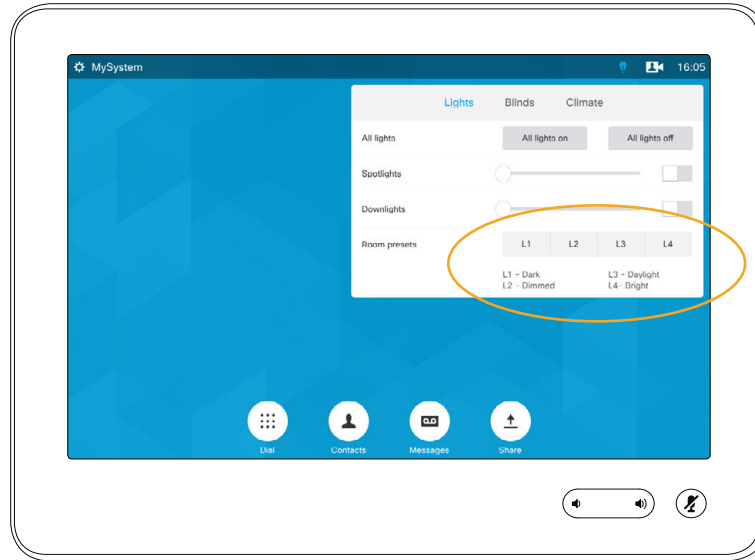
Video Switch Example

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.
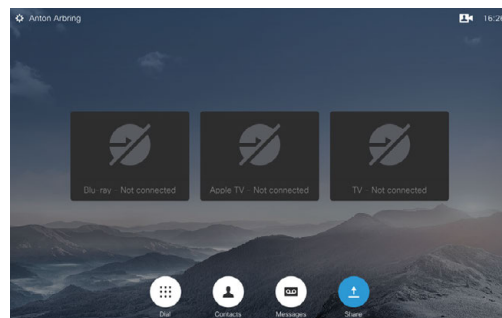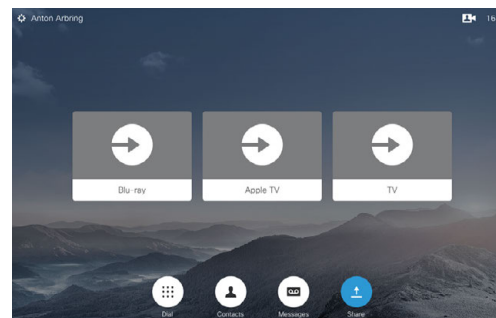
Fig. 1 Default state is NotReady.

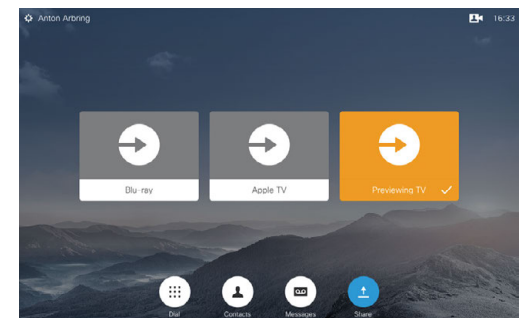Fig. 2 When Input sources have been set to Ready.

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

Simulator

A Sales Tool

CISCO

Running the Simulator

In order to make it possible to demonstrate the features of the In-Room Control, we have made a simulator for you. The simulator should primarily be regarded as a sales tool.

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).
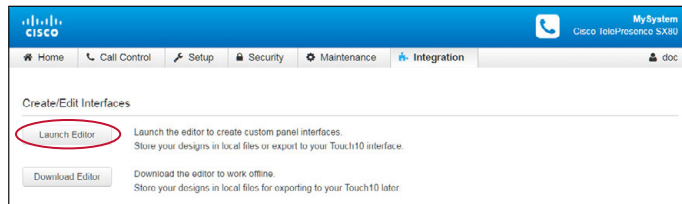
To avoid conflicts, any controller connected to the codec should be disconnected when running the simulator.

We recommend that you use Google Chrome browser when running the simulator. Other browsers may fail to run the simulator properly.

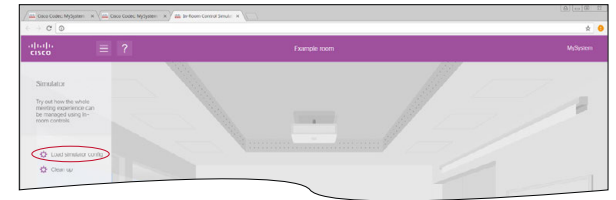To invoke the simulator navigate to *Integration* > *In-Room Control* in the web interface:
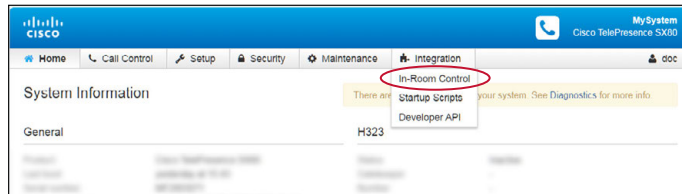
Select *Launch Editor*:

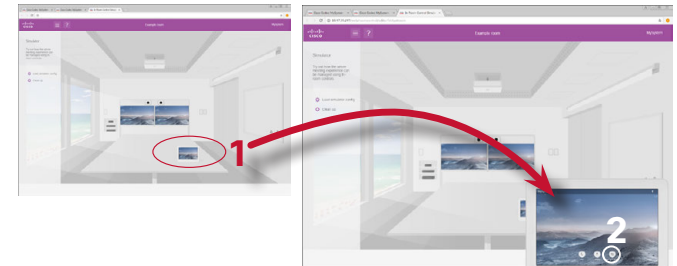
In the Editor, select *Simulator*:
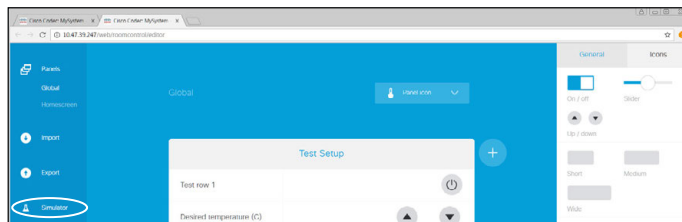
The Example Room will now appear. Click on *Load Simulator Config*. Once succesfully loaded click anywhere to remove messages when needed.

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

You may now play with the controls on the virtual Touch, but also with the switches themselves in the room. Click *Clean up* to finish.
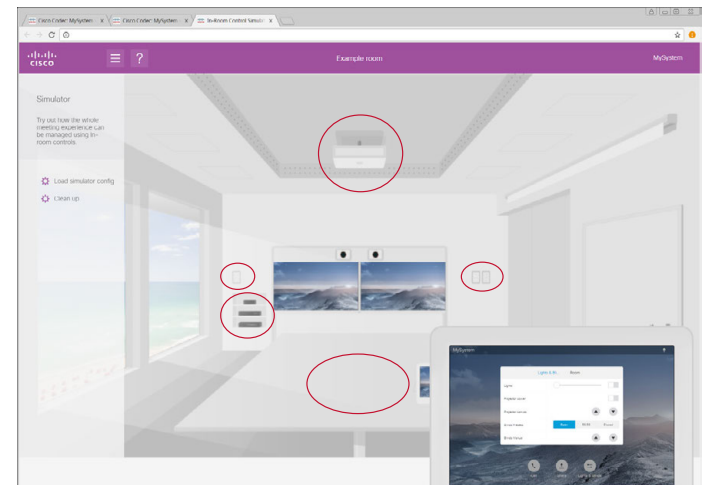
Cisco contacts

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

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED "AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

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.

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: www.cisco.com/go/trademarks. Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1110R)