カスタマイズガイド

Cisco Webex デバイス実行向け RoomOS 10.11

このユーザガイドは、ビデオ会議デバイスのセットアップと設定を操作する管理者向けに設計されています。

本書上部のメニュー バーと目次の各項目には、すべてハイパーリンクが設定されています。クリックすると、そのトピックに移動します。

Cisco TelePresence システムのユーザガイド、コンプライアンス、および安全性情報は、次のサイトから参照できます。

►https://www.cisco.com/go/telepresence/docs

Cisco Webex Cloud サービスに登録されたデバイスの詳細については、以下のサイトを参照してください。

►https://help.webex.com

次の Cisco ウェブ サイトに定期的にアクセスして、資料の最新バージョンを確認してください。

目次

第 1 章

はじめに

カスタマイズの機会

カスタマイズをサポートするビデオデバイスは次のとおりです。

- ▶ Cisco Webex Board シリーズ
- ▶ Cisco Webex Desk シリーズ
- ▶ Cisco Webex Room シリーズ

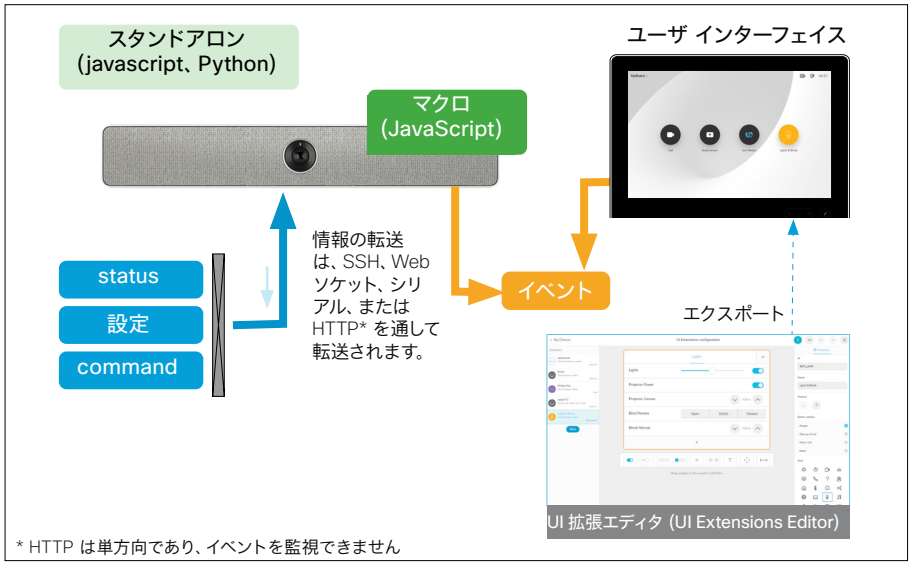
ローカル Web インターフェイスに加えて、次の方法でデバイス構成を変更できます。

- 端末モード (SSH)
- XML
- HTTP/HTTPS
- xAPI
- JSXAPI
- マクロ
- WebSocket

オープンなプラットフォームを使用すると、既存のワークフローに接続するカスタマイズを作成できます。したがって、中核的な Webex サービスを超えて追加の値を作成することができます。

以下に例を示します。

- チケットシステムへの問題報告
- Web サーバーからパネルに表示されるデータの取得
- 部屋から音響データを収集し、処理可能な Web サーバーに送信する
- 照明、ブラインド、ビデオスイッチなどのサードパーティ製周辺機器の制御



xAPI を通したプログラミング

用語の定義

*アクションボタン。*アクション ボタンは、ユーザインターフェイスに追加できる簡単なボタンです。押すと、定義済みの番号をダイヤルするなどのコマンドが実行されます。

*制御システム。*コントロールシステムは、周辺機器用のハードウェアドライバを備えたサードパーティ製システムです（例：Crestron、AMX、Raspberry Pi）。

*タッチ コントローラ*タッチコントローラは、Desk シリーズを除くすべての製品で使用されるシスコのタッチベースの制御デバイスを指します。タッチコントローラには、Cisco Touch 10 または Cisco Webex Room Navigator のいずれかを使用できます。

マクロマクロは、JavaScript および xAPI コマンドで作成された短いスクリプトです。詳細については、マクロセクションを参照してください。

マクロ エディタ (Macro Editor) **マクロ エディタ**は 、カスタム マクロを作成およびデバッグできるコード編集ツールです。詳細については、マクロ エディタの紹介 セクションを参照してください。

*パネル*パネルは、ユーザインターフェイスに追加できるカスタム作成の制御（ボタン、スライダ、スイッチなど）グループです。対応する制御アイコンをタッチすると、パネルが開きます。詳細については、パネルセクションを参照してください。

*ユーザーインターフェイス (UI)。*ユーザーインターフェイスとは、ビデオデバイスのタッチスクリーンを指します。タッチスクリーンは、Webex Board や Desk Pro などに組み込まれている場合もあれば、Room Navigator などの外付けの場合もあります。

*ユーザインターフェイス拡張機能エディタ***ユーザインターフェイス拡張機能エディタ**は使いやすく、ユーザインターフェイス上にカスタムボタンとパネルを作成できます。詳細については、UI 拡張機能エディタの紹介 セクションを参照してください。

*ビデオデバイス*ビデオデバイスとは、シスコビデオ会議システム（例：Webex Board、Desk Pro）のことを指します。

*Web アプリ。*Web アプリは、デバイス上にインターネットページを表示するアプリケーションです。これは、ユーザーインターフェイスにボタンを追加することでプログラムできます。ボタンを押すと、事前定義された Web サイト が起動します。

*ウィジェット。*ウィジェットは、パネル上の制御です。ウィジェットには、スイッチ、ボタン、スライダー、テキストなどが含まれる場合があります。詳細については、 ウィジェット (Widgets) セクションを参照してください。

*xAPI。*xAPI は、ビデオデバイスのアプリケーション プログラミング インターフェイス (API) です。xAPI は、ビデオデバイスとサードパーティ製アプリケーション間の通信を容易にします。詳細については、 アプリケーション プログラム インターフェイス (API) の章を参照してください。

第 2 章

ユーザーインターフェイス拡張機能

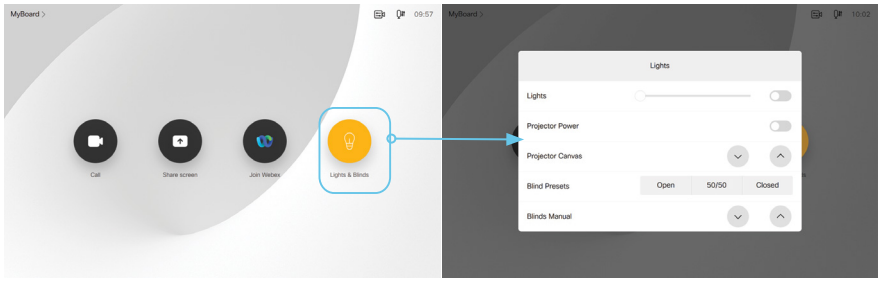
ユーザ インターフェイス拡張機能とは

Webex Series のビデオデバイスには、ビデオ会議デバイスのタッチスクリーンでツールを拡張できる**ユーザーインターフェイス (UI) 拡張機能エディタ**が含まれています。

ドラッグアンドドロップで簡単に操作できる制御エディタには、ウィジェットと呼ばれるユーザ インターフェイス要素のライブラリが用意されています。これらのウィジェットを使用して、独自の制御パネルを作成できます。

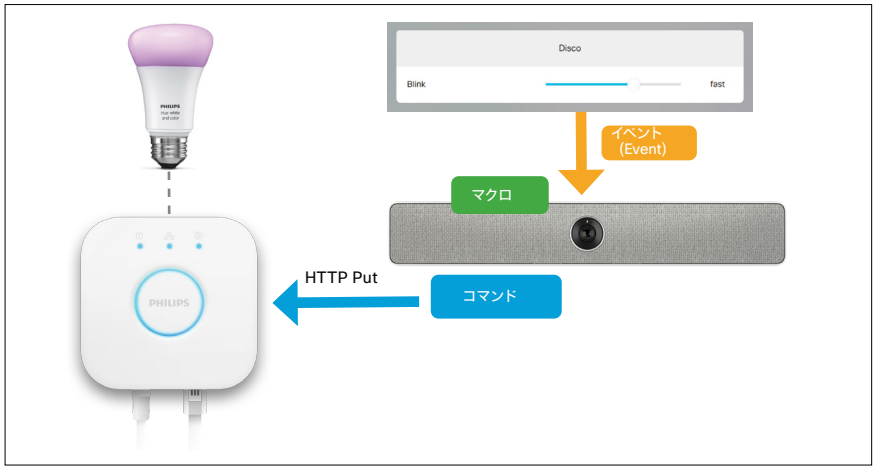
ボタンやパネルなどの UI 拡張機能は、無限の方法でプログラムできます。

たとえば、このように音楽を制御するためのカスタムパネルを作成できます。



ユーザインターフェイスのカスタムパネル

このように、光の点滅を制御するシステムをプログラムできます。



xAPI および HTTP によるカスタムアクション

ユーザが情報を送信できるテキストフィールドを含むカスタムパネルを作成できます。情報をサーバに送信し、保存、処理、表示できます。

この章では、各ウィジェットのプログラミングとモニタリングの簡単な例を説明します。例や オンラインリソース へのリンクは、例 章を参照してください。

UI 拡張機能の作成

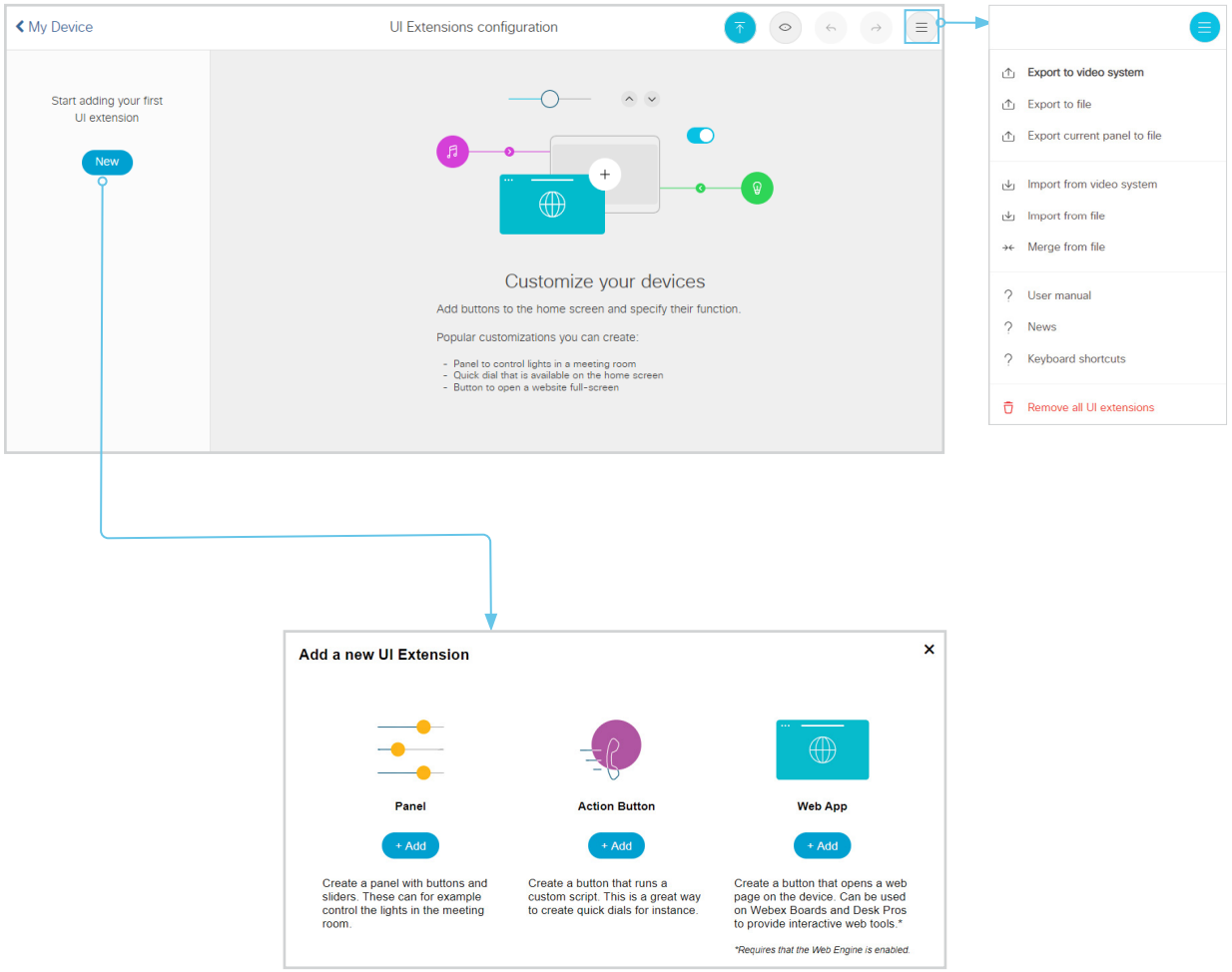
UI 拡張機能エディタにアクセスするには、ビデオデバイスの Web インターフェイスに *Administrator*、*RoomControl* または *Integrator* のログイン情報でサインインし、*[カスタム (Customization)] > [UI 拡張機能エディタ (UI Extensions Editor)]* の順に選択します。

[新規 (New)] をクリックすると、**[新規拡張機能を追加 (Add a new extension)]** ダイアログが表示されます。

以下のオプションが表示されます。

- パネル - 多数のウィジェット (スライダ、スイッチ、ボタンなど) を含む制御パネル。
- アクション ボタン - 押した時にコマンドを実行するシンプルなボタン (番号をダイヤルするなど) 。
- Web アプリ - Web Engine があるビデオデバイスの場合、ボタンは、ユーザーインターフェイスで全画面表示で Web ビューが起動します。

それぞれ、ユーザインターフェイスに新しいボタンを追加します。スペースがなくなる前に、メインページに追加されるボタンは少数のみです。オーバーフローボタンにアクセスするには、アイコンが表示される画面領域で右から左にスワイプします。

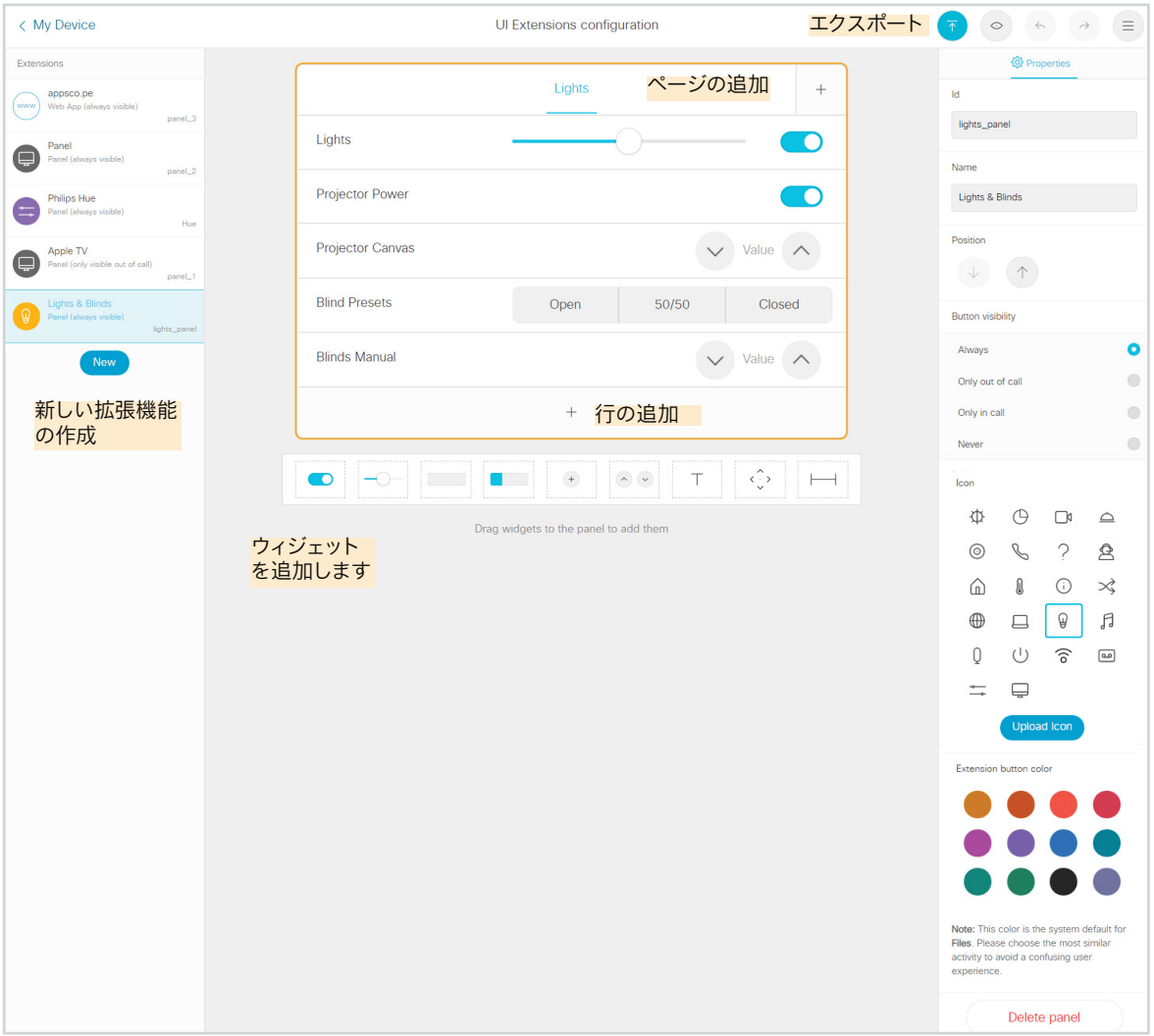


UI 拡張機能エディタの紹介 (1/5 ページ)

開始するには、新しいパネルを作成し、いくつかのウィジェットをパネルにドラッグします。

デフォルト名は、ダブルクリックして新しいテキストを入力して変更できます。*[確定 (Enter)]* をクリックして変更を適用します。

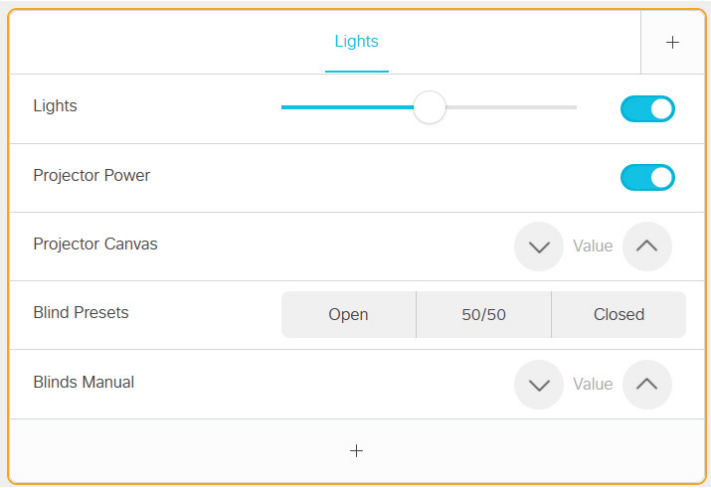
デバイス上で変更内容を確認する準備ができたら、↑をクリックします。

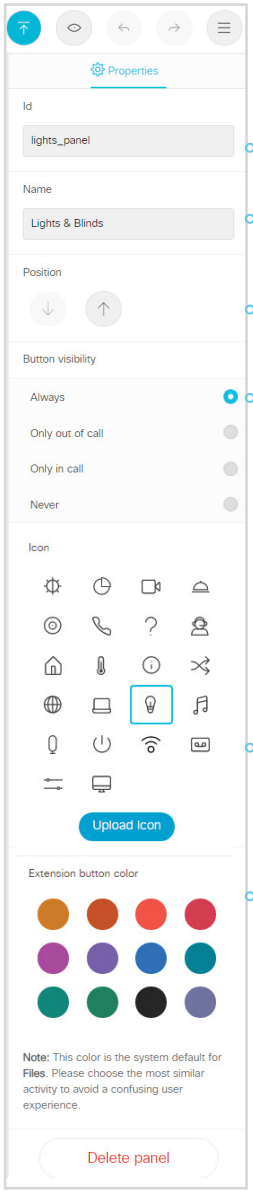


UI 拡張機能エディタの紹介 (2/5 ページ)

タイトルをクリックすると、パネルまたはウィジェットのプロパティにアクセスできます。

項目の周りに緑色のフレームが表示され、選択されている場合は **[プロパティ (Properties)]** ペイン に設定が表示されます。





[プロパティ (Properties)] ペインには次の設定が含まれます。

ID - 項目の識別子。これは API で、項目のコマンドまたはイベントの監視に使用されます。

名前 - このフィールドに入力したテキストがウィジェットの近くまたはパネル見出しに表示されます。

位置 - 位置矢印を使用して、ユーザーインターフェイスのボタンの順序を変更します。上向きの矢印は、順序の前にボタンを移動します。

ボタンの可視性 - このフィールドは、項目をいつ使用できるかを指定します。

- *Always* - 通話中と通話外の両方で使用可能
- *通話中のみ* - 通話外でのみ使用可能
- *通話中のみ* - 通話中のみ使用可能
- 非表示*にしない*

通話中にボタンを表示するには、次の操作を実行します。

- Boards：通話中に *[通話中のみ]* ボタンを表示するには、画面をタップします。通話中に*Always*ボタンを表示するには、*[ホーム (Home)]* ボタンをタップします。
- Desk Series デバイス - 通話中に画面をタップすると、ボタンが表示されます。

アイコン - アイコンはボタンに表示される画像です。デフォルトのアイコンのいずれかを選択するか、独自にアップロードします。

色 - ボタンの色。標準のボタンに使用される限定的なカラーパレットは、エディターで使用できます。色を選択すると、シスコでその色を使用する場合についての簡単な説明が表示されます。

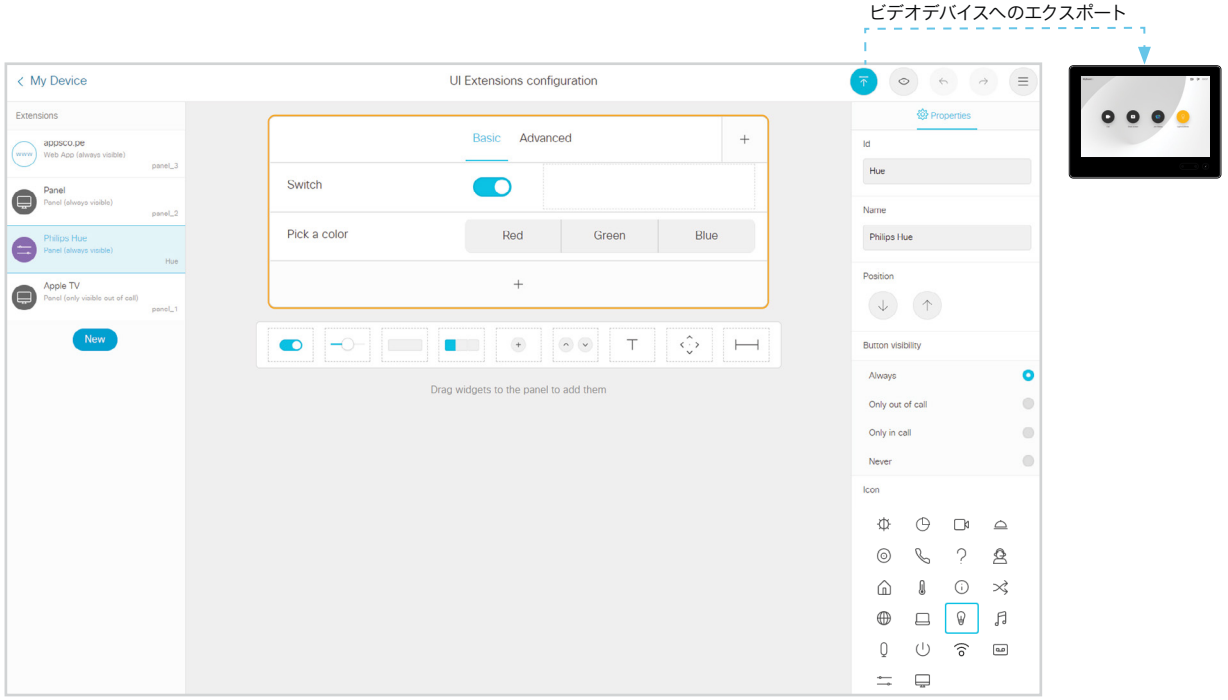
注：インコールのユーザインターフェイスのボタンには色を設定できません。これらのボタンは通常黒です。

UI 拡張機能エディタの紹介 (3/5 ページ)

エディタのヘッダー領域にはいくつかのボタンがあります。

- 変更をユーザインターフェイスにエクスポートします。
- 変更内容のプレビューを表示します。
- および [元に戻す] および [やり直し]
- その他のオプションを表示します。

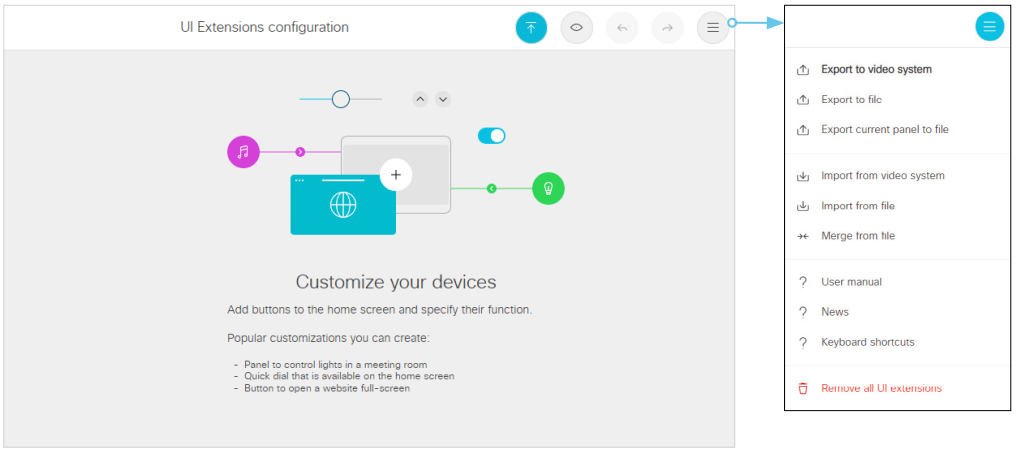
終了するには、左上隅の Cisco ロゴをクリックします。



UI 拡張機能エディタの紹介 (4/5 ページ)

UI 拡張機能エディタの右上隅のアイコンに、いくつかの重要なオプションが表示されます。

- *ビデオシステムへのエクスポート* - エディタからユーザインターフェイスに UI 拡張機能をエクスポートします。これにより、ビデオデバイスの既存のカスタム拡張機能が上書きされます。これは、ボタンをクリックした場合と同じ動作です。
- *ファイルへのエクスポート* - エディタから XML ファイルに UI 拡張機能をエクスポートします。
- *現在のパネルをファイルにエクスポート* - 現在選択しているパネルの設定だけを XML ファイルにエクスポートします。
- *ビデオシステムからのインポート* - ビデオデバイスのユーザインターフェイスの設定を取得し、エディタに適用します。エディタで保存されていない変更がある場合、これらは消去されます。
- *ファイルからのインポート* - オフライン設定を XML ファイルとしてインポートします。エディタで保存されていない変更がある場合、これらは消去されます。
- *ファイルからの統合* - オフライン設定を XML ファイルとしてインポートし、エディタの現在の設定に追加します。注：同じ名前の任意のパネルは、上書きされます。
- *ユーザマニュアル* - 複数のバージョンのユーザマニュアルへのリンクを含む Web ページを開きます。
- *[ニュース (News)]* - 最近のリリースからの変更点に関する情報を参照します。



- *キーボードショートカット* - エディタで使用する一般的なキーボードショートカットの一覧を参照します。Mac ユーザの場合は、`Ctrl` を `Cmd` に置き換えます。

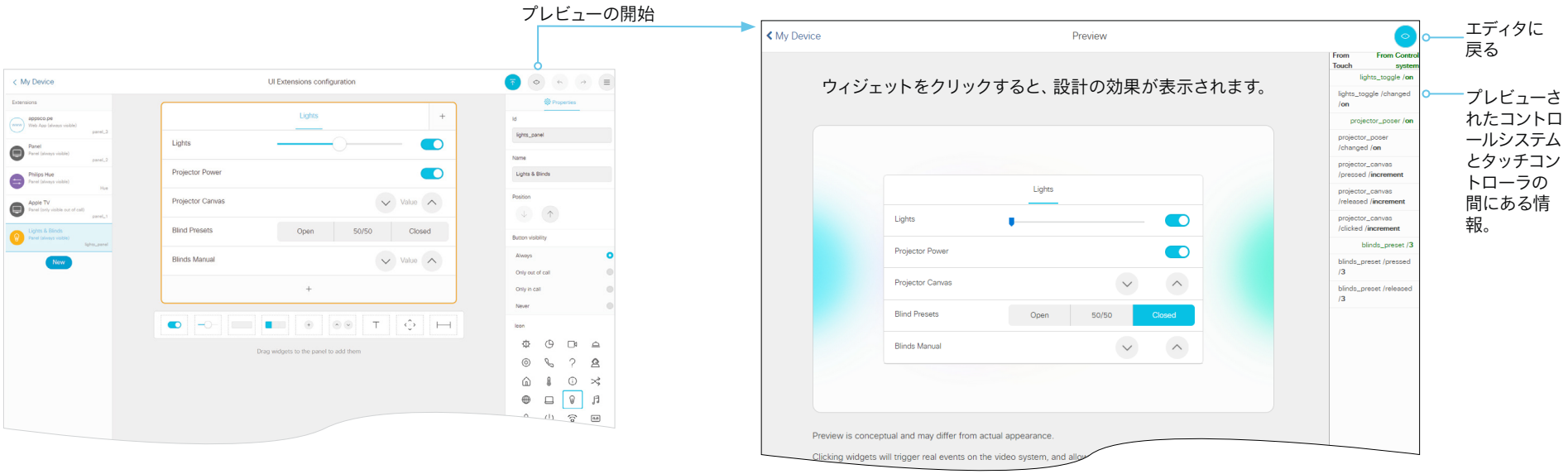
Ctrl-Enter	ビデオシステムへのエクスポート構成
Ctrl-Space	現在の設定のプレビュー
Ctrl-S	設定をファイルに保存
Ctrl-O	ファイルから設定を開く
Ctrl-Z	前回の操作を元に戻す
Ctrl-Shift-Z	前回の操作をやり直す
Ctrl-Shift-C	選択したコンポーネントのコピー
Ctrl-Shift-X	選択したコンポーネントの切り取り
Ctrl-Shift-V	選択したコンポーネントの貼り付け
Ctrl-D	選択したコンポーネントの複製
Backspace / Delete	選択したコンポーネントの削除

- *すべての UI 機能拡張を削除* - エディタが消去されますが、ビデオデバイスは消去されません。この変更を元に戻すには、*ビデオシステム からインポート*を選択します。この変更をビデオシステムにプッシュするには、*[ビデオシステムにエクスポート (Export to video system)]* を選択します。

メニューを閉じるには、メニューアイコンをクリックします。

UI 拡張機能エディタの紹介 (5/5 ページ)

設定を展開する前に、プレビューして確認することができます。



注: プレビューは、すべてのデバイスで機能しますが、タッチコントローラ用にすべてが作成されているのと同じ方法で表示されます。

上記の図は、シミュレートされたサードパーティ製のコントロールシステムが接続されている、シミュレートされたビューを示しています。

設定を実装する際 (実際の状況シナリオ)、可能な場合は必ず制御システムが `SetValue` コマンドを送信するように設定してください。

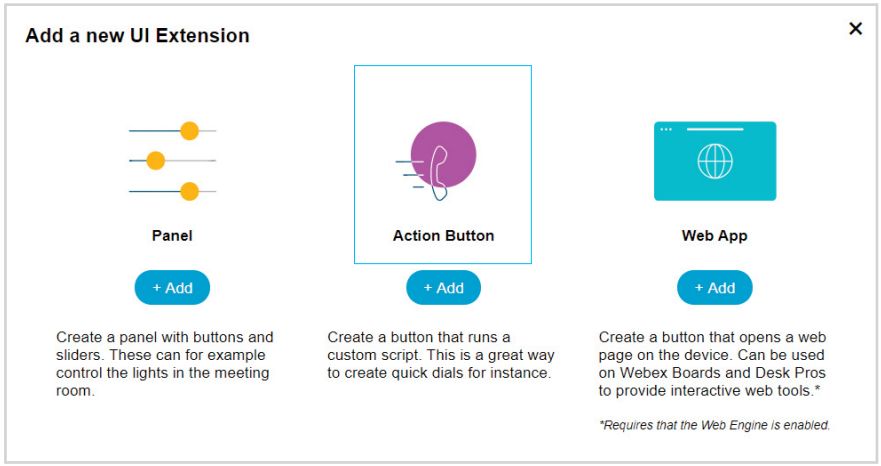
例: 実際の状況で [照明 (Lights)] を [オン (On)] に設定する場合、タッチコントローラは照明が実際にオンにされたことを確認するフィードバックを受け取る必要があります。これが行われるようにするには、コントローラは照明をオンにしてから `SetValue` を送信して、照明の設定が変更されたことを確認する必要があります。上記の例の右側のペインには、タッチコントローラがコントロールシステムに送信した内容と、それに対してコントロールシステムがタッチコントローラに返信した内容を示すシミュレーションが示されています。

実際の状況では、誰かが会議室の壁の照明スイッチを操作したときにはいつでも、コントロールシステムが `SetValue` タッチコントローラに送信していることを確認する必要があります。

アクション ボタン

アクションボタンを作成する場合は、ユーザインターフェイスにボタンを追加します。ユーザがボタンを押すと、1 つのアクション（番号のダイヤルなど）が実行されます。

API コマンドを使用して、このアクションをプログラムする必要があります。**マクロエディタ**を使用してこのプログラムを作成する方法は簡単です 。詳細については、マクロの章も参照してください。



アイコンの色と画像や画面の位置も選択できます。 ボタンをクリックしてボタンをエクスポートし、ユーザインターフェイスで表示します。

アクションボタンの例

次の手順では、メッセージを表示するアクションボタンを作成します。

1. 「hello1_button」（引用符なし）の ID を使用してアクションボタンを作成します。
2. ボタンをクリックすると、アクションボタンがビデオデバイスにエクスポートされます。

   ユーザインターフェイスを確認します。アクションボタンが表示されます。ボタンがプログラムされていないため、押した場合は何も起こりません。
3. ボタンを押した時に実行するコマンドをセットアップする必要があります。

   マクロエディタを開いて、*［新しいマクロの作成 (Create new macro)*をクリックします。テキストエディタが開きます。最初のテキストの後に次のスクリプトを追加します。

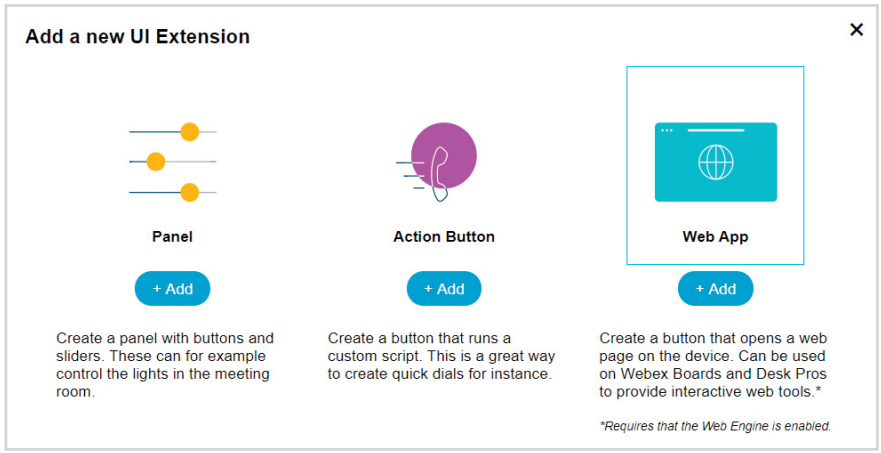
```
xapi.event.on('UserInterface Extensions Panel Clicked', (event) => {
  if (event.PanelId === 'hello1 _ button') {
    xapi.command('UserInterface Message Prompt Display', { Title: 'Hello!',
text: 'Have a great day!'} );
  }
});
```

4. `Ctrl+s` と入力してマクロを保存します。
5. マクロを有効化します。これは、対応するスイッチの設定 を *On* にすることで実行できます。
6. ビデオデバイスで、ボタンをテストします。フレンドリーなメッセージが表示されます。

Web アプリ

Web アプリ は、ホームページ上のシンプルなボタンです。ユーザがこのボタンを押すと、Web ビューが全画面で起動します。

注：Web アプリは、Web Engine があるデバイスのみで使用されます (例：Webex Board や Desk Pro) 。Web アプリを作成するには、Web Engine を有効にする必要があります。

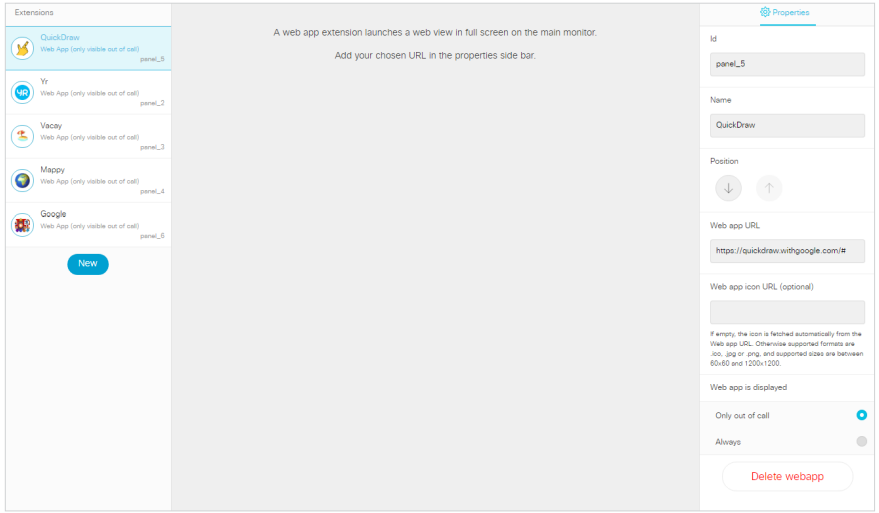


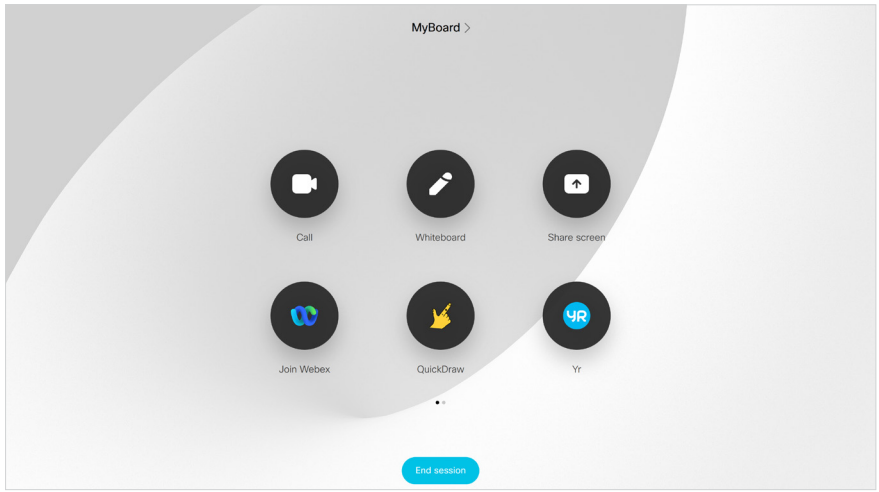
注：Webex Board、Web アプリは、通話外でのみ使用できます。

Web アプリの作成

1. ボタンの下に表示する名前を入力します (Twitter など) 。
2. Web サイトの URL (twitter.com など) 。
3. アイコンの色と画像を指定します。
4. または、Web アプリアイコンの URL を指定して Favicon を取得します。

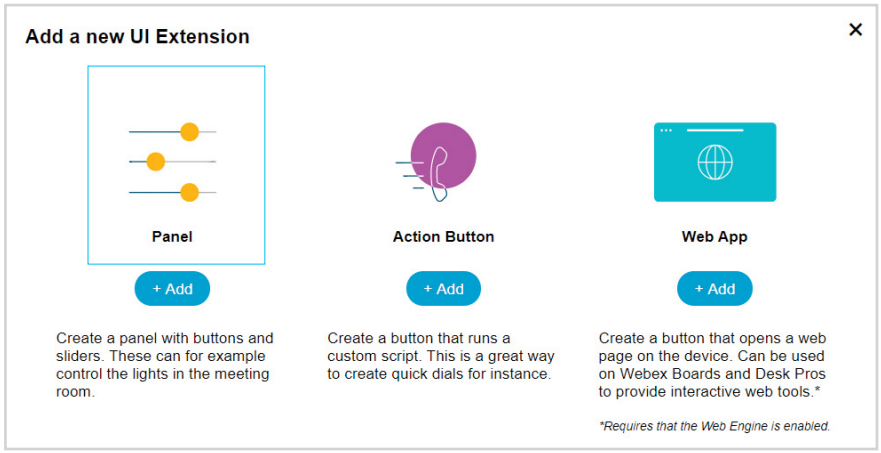
   Favicon はエディタにもプレビューにも表示されないことに注意してください。これは、左の下の画像が (例として Twitter を使用して) ビデオデバイスに表示されます。
5. ボタンをクリックして、Web アプリをエクスポートし、ユーザーインターフェイスで表示します。





パネル

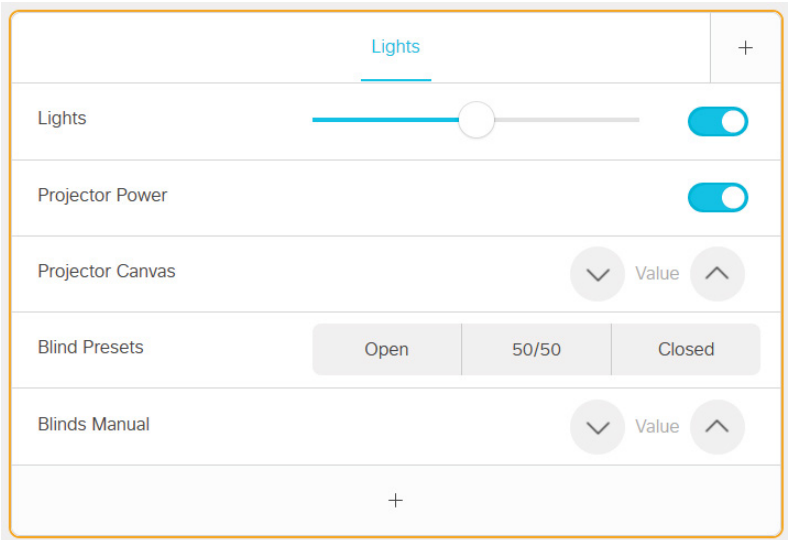
パネルを作成する場合は、ユーザインターフェイスにボタンを追加します。ユーザがボタンを押すと、パネルが開きます。



パネルには通常、ボタン、スイッチ、スライダーなど、複数のウィジェットがあります。詳細については、「ウィジェット (Widgets)」セクションに説明があります。

パネルのウィジェットは、次のガイドラインに従って 4 列の枠線に配置されます。

- 行は右揃えになります。
- ウィジェットは、サイズに応じて 1 ~ 4 列を埋めます。
- 1 行に収まらない数のウィジェットを追加すると、ウィジェットは同じ行内で折り返されます。



関連する制御を同じページでグループ化すると良いです。さらにページをパネルに追加するには、パネルの右上にある [+] アイコンをクリックします。

ページは、ウィジェットに入力できる 1 つ以上の行で構成されています。

作成したページが、パネルに個別のタブとして表示されます。パネル 1 つで最大 50 ページまで使用できます。

パネル数は無制限ですが、各ページに表示されるボタンは数ボタンのみです。オーバーフローボタンにアクセスするには、アイコンが表示される画面領域で右から左にスワイプします。

ユーザがパネルを操作した場合の対応は、ユーザが管理します。これらのウィジェットが相互対話を監視し、必要なコマンドを実行するプログラムをセットアップする必要があります。マクロはこれに関して非常に便利です。マクロの作成方法については、マクロ章を参照してください。

ウィジェット (Widgets) (1/17ページ)

ウィジェットについて

パネルは、「ウィジェット」と呼ばれるユーザインターフェイス要素で構成されています。

ウィジェットのタイプには、次のようなものがあります。

- スイッチ
- スライダ
- ボタン
- グループ ボタン
- アイコン ボタン
- スピナー
- [テキスト (Text)]
- 方向パッド
- スペーサ

ウィジェットの動作は、API または XML を介してプログラムされます。これについての詳細は、次のページで重点を置いて説明されています。

- ウィジェットの値を変更するコマンド。
- 送信されるイベント
  (Pressed、Changed、Released、Clicked) と、これらのイベントをトリガーするアクション。
- マクロ、端末出力モード、または XML 出力モードでのコマンドとイベントの例。

ユーザインターフェイス拡張機能に関するすべてのイベント、コマンド、およびステータスの構文とセマンティクスは、「アプリケーション プログラム インターフェイス (API) 」の章で説明されています。

API は、端末、XML、API、または**マクロエディタ**から使用できます。マクロの作成方法については、マクロ章を参照してください。

ウィジェットの識別子

パネル上のすべてのウィジェットには、一意の識別子であるウィジェット ID が必要です。ウィジェット ID はユーザが定義するか、自動的に割り当てることができます。ウィジェット ID に任意の名前か番号を指定できます。特殊文字は含めずにわかりやすい名前を使用することをお勧めします。最大文字数は 255 です。

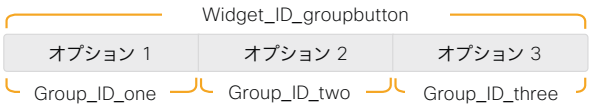
ウィジェット ID は、ユーザーインターフェイス、ビデオデバイス、およびコントロールシステム間のプログラミングリンクです。ウィジェット ID は、ウィジェットに関連付けられているすべてのイベントに含まれます。コードを介してそのウィジェットにコマンドを送信する場合は、同じ識別子を使用する必要があります。

グループの識別子

ウィジェットの 1 つ 、[Group] ボタン には、次の 2 種類の識別子があります。

- ウィジェット ID - ボタンの完全なグループを指します。
- グループ ID - グループ内の個々のボタンの一意の識別子。

  グループ ID は自動的に割り当てられますが、変更は可能です。グループ ID には任意の名前や番号を使用できます。特殊文字を含めずにわかりやすい名前を使用することをお勧めします。最大文字数は 255 です。



イベントとコマンド

イベントは、発生した事についての通知です (「照明スイッチがオフになっている」など)。

コマンドは、何かが起こるようにリクエストするものです (「照明を消す」など) 。

任意の UI 拡張機能のステータスが変更された場合 (つまり、オン/オフ) 、イベントが送信されます。これらのイベントを受信し、ご希望に基づいてコマンドを送信するよう登録することができます。

たとえば、部屋の照明を制御するスイッチを作成する場合は、スイッチが *[オン(On)]* から *[オフ(Off)]* にされた場合とその逆も検出する必要があります。

照明スイッチが変更されたイベントを検出したら、物理的な照明のオン/オフを切り替えるコマンドを送信します。

同様に、他の制御メカニズムでライトをオン/オフした場合も、このステータスの変化を検出してから、スイッチのパネルの見た目を変更する必要があります。これは、ウィジェットの値を設定する `Set Value` コマンド で行います。

ウィジェット (2/17ページ)

スイッチ (1/2 ページ)

スイッチは 2 つの状態の制御で、オンまたはオフのいずれかを示します。

使用例 : オンまたはオフにできる機能 (照明、ファン、プロジェクタなど) 。

明るさを調整する照明には、スライダと併用するトグル ボタンとして使用することもできます。

コマンド

`SetValue` を使用してパネルの表示位置を変更するコマンドをスイッチに送信します 。

例 : myswitch の WidgetId を使用してスイッチをオンにします。

端末モード

たとえば、端末で経由してスイッチをオンにする場合は、次の操作を実行します。

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

マクロ

たとえば、マクロ経由で「myswitch」の WidgetId を使用してスイッチをオンにする場合は、次の操作を実行します。

```
xapi.command('UserInterface Extensions Widget SetValue', {WidgetId:
'myswitch', value: 'on'});
```

ウィジェット (3/17ページ)

スイッチ (2/2 ページ)

イベント

UI で変更を監視します。スイッチをタッチしてステータスを変更すると、次のイベントが発行されます。

- *[オン(On)]* - スイッチが「オフ」から「オン」に切り替えるとトリガーされます。値："on"
- *[オフ(Off)]* - スイッチが「オン」から「オフ」に切り替えるとトリガーされます。値："off"

例：「myswitch」の WidgetId を使用してボタンを押して放します。端末、XML、またはマクロを使用して、このアクションを検出します。

端末モード

端末を介したウィジェットのイベントに登録します。例：

```
xFeedback Register Event/UserInterface/Extensions/Widget/Action
```

スイッチを変更すると、イベントがブロードキャストされます。例：

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

XML モード

たとえば、XML は次のようになります。

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

マクロ

マクロを介してイベントを監視するには、次のような例があります。

```
xapi.event.on('UserInterface Extensions Widget Action', (event) => {
    if ((event.WidgetId === 'myswitch') &&
        (event.Type === 'changed')) {
      console.log(event.WidgetId, 'changed to',
                  event.Value);
    }
  });
```

出力：

```
'myswitch' 'changed to' 'on'
```

ウィジェット (4/17ページ)

スライダ (1/2 ページ)

スライダを使用すると、設定範囲内の値を選択できます。

最小値は 0、最大値は 255 で表されます。スライダを押したまま移動させると、1 秒間に最大 5 回のイベントが送信されます。

バーをタップすると、スライダはすぐにその新しい位置に移動します。

使用例 : 明るさを調整できる照明、音量コントロール。

コマンド

パネルのスライダの位置を更新するコマンドを送信します。

例 : 「mys slider」の WidgetId を使用してスライダを「98」に設定します。

端末モード

端末でコマンドを送信するには、次のような例があります。

```
xCommand UserInterface Extensions Widget SetValue WidgetId "myslider" Value:
"98"
```

マクロ

マクロでコマンドを送信するには、次のような例があります。

```
xapi.command('UserInterface Extensions Widget SetValue', {WidgetId:
'myslider', value: '98'});
```

ウィジェット (5/17ページ)

スライダ (2/2 ページ)

イベント

UI で変更を監視します。スライダを動かしたりクリックしたりすると、次のイベントが発生します。

- *Pressed* スライダを押すとトリガーされます。値 : 該当なし
- *Changed* - スライダを押したまま移動させて、スライダを離したときにトリガーされます。
  値 : 0 ～ 255
- *Released* - スライダを離すとトリガーされます。
  値 : 0 ～ 255

例 : "myslider" の WidgetId とスライダを押したまま新しい位置 ("194") に移動させて、スライダを離します。端末およびマクロを介して、このアクションを検出します。

端末モード

端末を介したウィジェットのイベントに登録します。例 :

```
xFeedback Register Event/UserInterface/Extensions/Widget/Action
```

スライダを移動すると、いくつかのイベントがブロードキャストされます。スライダを放すと、値を設定できます。例 :

```
*e UserInterface Extensions Widget Action WidgetId: "myslider"
*e UserInterface Extensions Widget Action Value: "194"
*e UserInterface Extensions Widget Action Type: "pressed"
** end
*e UserInterface Extensions Widget Action WidgetId: "myslider"
*e UserInterface Extensions Widget Action Value: "194"
*e UserInterface Extensions Widget Action Type: "changed"
** end
*e UserInterface Extensions Widget Action WidgetId: "myslider"
*e UserInterface Extensions Widget Action Value: "194"
*e UserInterface Extensions Widget Action Type: "released"
```

XML モード

たとえば、XML は次のようになります。

```
<Event>
  <UserInterface item="1">
    <Extensions item="1">
      <Widget item="1">
        <Action item="1">
          <WidgetId item="1">myslider</WidgetId>
          <Value item="1">194</Value>
          <Type item="1">released</Type>
        </Action>
      </Widget>
    </Extensions>
  </UserInterface>
</Event>
```

マクロ

マクロを介してイベントを監視するには、次のような例があります。

```
xapi.event.on('UserInterface Extensions Widget Action', (event) => {
    if ((event.WidgetId === 'myslider') && (event.Type === 'released')) {
         console.log(event.WidgetId, 'changed to', event.Value);
    }
  });
```

出力 :

```
'myslider' 'changed to' '194'
```

ウィジェット (6/17ページ)

ボタン (1/2 ページ)

ボタンは、通話などのアクションの実行、または何が On または Off かどうかを示すのに使用できます。

2 回使用する場合は、ボタンの値を "active" または "inactive" に変更するコマンドを送信する必要があります。このコマンドにより、その色も変更されます。その後、ユーザが再度押した場合は、必ず "inactive" に戻してください。

エディタ内でボタンをダブルクリックして、テキストを変更します。カスタム テキスト付きのボタンは、さまざまなサイズで配置できます。ボタンのサイズによって、追加できる最大文字数が決まります。テキストは次の行に折り返されません。

`SetValue` コマンドを使用して、テキストを動的に変更することはできません。

同時に 1 つのボタンのみを選択できるようにリンクされた複数のボタン (ラジオ ボタンなど) が必要な場合は、グループ ボタンを使用することを検討してください。

注意 : これらは アクション ボタン と同じではありません。よって、別のコマンド一式が必要です。

使用例 : 対象のオンとオフを切り替えます。

コマンド

ユーザインターフェイスのボタンをハイライトするには、`SetValue` コマンドを使用します。値が "active" の場合はボタンがハイライトされ、値が "inactive" の場合はハイライトが解除されます。

例 : 「mybutton」の WidgetId を持つボタンをアクティブな状態に設定します。

端末モード

端末経由で「mybutton」の WidgetId を使用してボタンをアクティブにするには、次の操作を実行します。

```
xCommand UserInterface Extensions Widget SetValue WidgetId: "mybutton" Value:
"active"
```

その後、非アクティブに戻すには、次の操作を実行します。

```
xCommand UserInterface Extensions Widget SetValue WidgetId: "mybutton" Value:
"inactive"
```

マクロ

マクロ経由で「mybutton」の WidgetId を使用してボタンをアクティブにするには、

```
xapi.command('UserInterface Extensions Widget SetValue', {WidgetId: 'mybutton',
value: 'active'});
```

その後、非アクティブモードに戻すには、次の操作を実行します。

```
xapi.command('UserInterface Extensions Widget SetValue', {WidgetId: 'mybutton',
value: 'inactive'});
```

ウィジェット (7/17ページ)

ボタン (2/2 ページ)

イベント

ボタンを押すと、次のイベントが発行されます。

- *Pressed* - ボタンを押すとトリガーされます。値 : 該当なし
- *Released* - ボタンを離すとトリガーされます。値 : 該当なし
- *Clicked* - ボタンを離すとトリガーされます。値 : 該当なし

例 : "mybutton" の WidgetId を使用してボタンを押して離し、端末やマクロを通じて このアクションを検出します。

端末モード

端末を介したウィジェットのイベントに登録します。例 :

```
xFeedback Register Event/UserInterface/Extensions/Widget/Action
```

ボタンを押して離すと、いくつかのイベントがブロードキャストされます。例 :

```
*e UserInterface Extensions Widget Action Type: "pressed"
*e UserInterface Extensions Widget Action Type: "released"
*e UserInterface Extensions Widget Action Type: "clicked"
```

XML モード

たとえば、XML は次のようになります。

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

マクロ

ボタンを検出して何らかの操作を実行するには、次の手順を実行します。

```
xapi.event.on('UserInterface Extensions Widget Action', (event) => {
    if ((event.WidgetId === 'mybutton') &&
        (event.Type === 'clicked')) {
          xapi.command('UserInterface Message Prompt Display',
                       { Title: 'こんにちは!', text: '良い一日を!'});
    }
  });
```

画面にメッセージが表示されます。

ウィジェット (8/17ページ)

グループボタン (1/2 ページ)

グループボタンがリンクされ、一度に選択できるのは 1 つのみです。

グループボタンは、ボタンをリンクするのに最適です。一度に 1 つだけを選択できます (ルームのプリセット など) 。グループ内の個々のボタンが小さすぎて説明テキストを含めない場合、説明にテキストウィジェットを使用できます。

グループボタンは、マトリックスまたは回線として作成できます。最大で 4 つのボタンに制限されません。マトリックスは最大 4 列と必要な数の行で構成されます。

まず、マトリックスに含める列の数 (1、2、3、または 4) を定義することから開始します。これはマトリックス全体 (すべての行など) に適用されるグローバル設定であり、1 行あたりのボタンの最大数を定義します。

ただし、行は最大数より少ないボタンを含むことが可能です。その後にボタンの自動サイジングが行われます。ボタンはつねに利用可能なスペースに収まります。

例 : 3 列のマトリックスを定義し、ボタンが 7 つ必要だとします (例として 3 行) 。システムはその後、ボタン 3 つを最初の行に、次の 3 つを 2 番目の行に、そして最後のボタンを 3 番目の行に配置します。3 番目の行に配置される 1 つのボタンは、スペースに収まるように (3 列をスパンするように) 自動サイジングされます。

ボタンのサイズによって、追加できる最大文字数が決まります。テキストは次の行に折り返されません。

`SetValue` コマンドを使用して、テキストを動的に変更することはできません。

使用例 : 互いに排他的なルーム プリセット。たとえば、[暗い (Dark)]、[クール (Cool)]、[明るい (Bright)] から選択できるルーム プリセット。プリセットが有効でなくなった場合 (壁掛け制御で照明を変更する場合など) 、プリセットの選択を解除 (リリース) するようにしてください。

もう 1 つの使用例 : UI の言語を違ったものに変更します。

コマンド

ボタンの外観は、UI でタップするとすぐに変化します。ただし、他の場所で行われた変更については、コントロールシステムは、ボタンのいずれかをタップした場合、常にビデオデバイスに `SetValue` コマンド を送信します。これにより、ステータスが確実に更新されます。

設定する値は、アクティブにするボタンの ID に対応しています。

`UnSetValue` コマンドを使用すると、グループ内のすべてのボタンを解放して、どのボタンもハイライトしていない状態にできます。

例 : 「mygroup」の WidgetId を持つグループボタンがあるという想定です。グループ内には、次のように ID を含むボタンが 3 つあります。{"high", "medium", "low"}. 端末とマクロで個別のボタンを選択します。

端末モード

id "high" のボタンを選択するには、次のコマンドを使用します。

```
xCommand UserInterface Extensions Widget SetValue WidgetId: "mygroup" Value:
"high"
```

次に、すべてのボタンを離します。

```
xCommand UserInterface Extensions Widget UnSetValue WidgetId: "mygroup"
```

マクロ

id="medium" のボタンを選択するには、次のコマンドを使用します。

```
xapi.command('UserInterface Extensions Widget SetValue', {WidgetId: 'mygroup',
value: 'medium'});
```

次に、すべてのボタンを離します。

```
xapi.command('UserInterface Extensions Widget UnSetValue', {WidgetId:
'groupbutton'});
```

ウィジェット (9/17ページ)

グループボタン (2/2 ページ)

イベント

UI で変更を監視します。グループボタンには、次のタイプのイベントが発行されます。

- *Pressed*いずれかのボタンを押すとトリガーされます。値：押されたボタン (グループ内) のグループ ID。
- *Released*いずれかのボタンを離すとトリガーされます。値：離したボタン (グループ内) のグループ ID。

例：「mygroup」の WidgetId を持つグループボタンがあるという想定です。グループ内には、次のように ID を含むボタンがあります。{"high", "medium", "low"}. ユーザインターフェイスで個々のボタンを選択し、端末やマクロで検出します。

端末モード

端末を介したウィジェットのイベントに登録します。例：

```
xFeedback Register Event/UserInterface/Extensions/Widget/Action
```

ボタンを押して離すと、いくつかのイベントがブロードキャストされます。例：

```
*e UserInterface Extensions Widget Action WidgetId: "mygroup"
*e UserInterface Extensions Widget Action Value: "high"
*e UserInterface Extensions Widget Action Type: "pressed"
** end
*e UserInterface Extensions Widget Action WidgetId: "mygroup"
*e UserInterface Extensions Widget Action Value: "high"
*e UserInterface Extensions Widget Action Type: "released"
** end
```

XML モード

たとえば、XML は次のようになります。

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

マクロ

マクロを使用してグループボタンの押下を検出するには、例：

```
xapi.event.on('UserInterface Extensions Widget Action', (event) => {
    if ((event.WidgetId === 'mygroup') &&
        (event.Type === 'released')) {
          console.log(event.WidgetId, 'changed to',
                  event.Value);
    }
  });
```

出力例：

```
'mygroup' 'changed to' 'low'
```

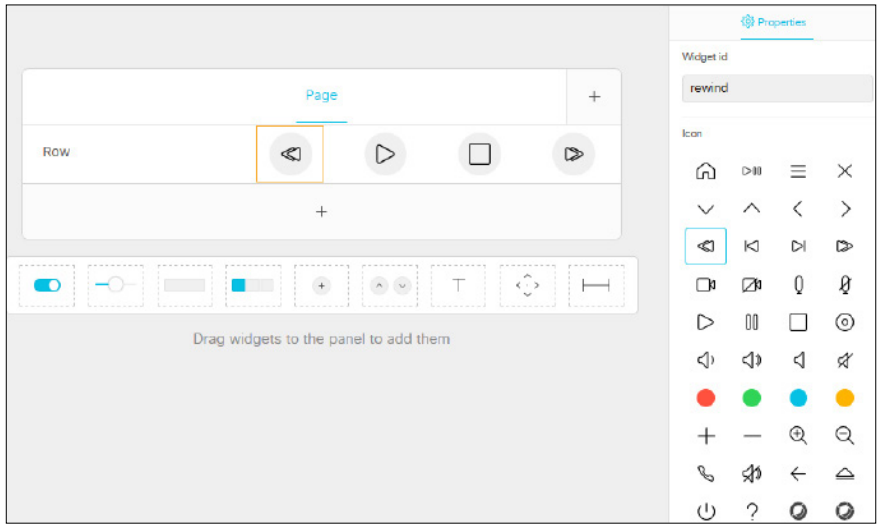
ウィジェット (10/17ページ)

アイコンボタン(1/2 ページ)

アイコンボタンは 、動作を他のボタンと共有しますが、アイコンには user-selected の色またはイメージがあります。

アイコンボタンには、active と inactive という 2 種類の状態があります。ユーザがボタンをタップしたときに、そのボタンを active 状態に設定する必要はありません。ボタンは、外観を変更せずに信号を送信するためだけに使用できます。

使用例 : メディア プレーヤーや、開始、停止、一時停止できるその他のデバイスを制御します。



コマンド

ユーザインターフェイスのボタンをハイライトするには、`SetValue` コマンドを使用します。値が "active" の場合はボタンがハイライトされ、値が "inactive" の場合はハイライトが解除されます。

例 : パネルに次の ID を持つ 4 つのアイコンボタンがあると想定します。{"rewind", "play", "stop", and "fast_forward"}. マクロと端末を介した "play" の WidgetId を持つボタンを選択します。

端末モード

"high" の WidgetId を持つボタンを選択するには、次のコマンドを使用します。

```
xCommand UserInterface Extensions Widget SetValue WidgetId: "play" Value:
"active"
```

その後、ボタンを離すには、次のコマンドを使用します。

```
xCommand UserInterface Extensions Widget SetValue WidgetId: "play" Value:
"inactive"
```

マクロ

"high" の WidgetId を持つボタンを選択するには、次のコマンドを使用します。

```
xapi.command('UserInterface Extensions Widget SetValue', {WidgetId: 'play',
value: 'active'});
```

その後、ボタンを離すには、次のコマンドを使用します。

```
xapi.command('UserInterface Extensions Widget SetValue', {WidgetId: 'play',
value: 'inactive'});
```

ウィジェット (11/17ページ)

アイコンボタン (2/2 ページ)

イベント

UI で変更を監視します。アイコンボタンには、次のタイプのイベントが発行されます。

- *Pressed* - ボタンを押すとトリガーされます。値:該当なし
- *Released* - ボタンを離すとトリガーされます。値:該当なし
- *Clicked* - ボタンを離すとトリガーされます。値:該当なし

例: パネルに次の ID を持つ 4 つのアイコンボタンがあると想定します。{"rewind", "play", "stop", and "fast_forward"}. ユーザインターフェイスを介して "play" の WidgetId を持つボタンを押し、端末やマクロ経由でこれらを検出します。

端末モード

端末を介したウィジェットのイベントに登録します。例:

```
xFeedback Register Event/UserInterface/Extensions/Widget/Action
```

ボタンを押して離すと、いくつかのイベントがブロードキャストされます。例:

```
*e UserInterface Extensions Widget Action WidgetId: "play"
*e UserInterface Extensions Widget Action Value: ""
*e UserInterface Extensions Widget Action Type: "pressed"
** end
*e UserInterface Extensions Widget Action WidgetId: "play"
*e UserInterface Extensions Widget Action Value: ""
*e UserInterface Extensions Widget Action Type: "released"
** end
*e UserInterface Extensions Widget Action WidgetId: "play"
*e UserInterface Extensions Widget Action Value: ""
*e UserInterface Extensions Widget Action Type: "clicked"
** end
```

XML モード

たとえば、XML は次のようになります。

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

マクロ

マクロを使用してグループボタンの押下を検出するには、例:

```
xapi.event.on('UserInterface Extensions Widget Action', (event) => {
    if ((event.WidgetId === 'play') &&
        (event.Type === 'clicked')) {
          console.log(event.WidgetId, 'was clicked.');
    }
  });
```

出力例:

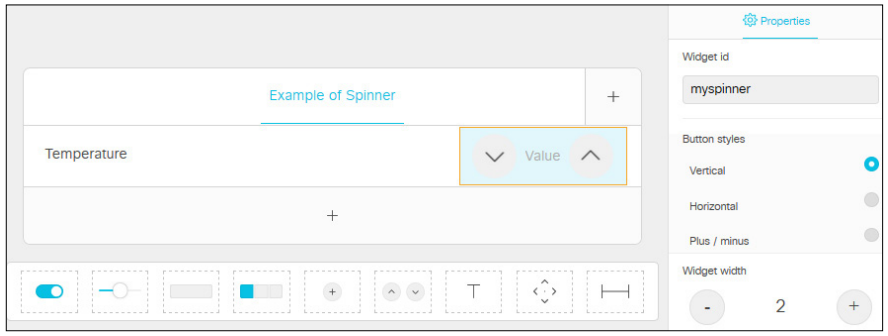
```
「再生」がクリックされました
```

ウィジェット (12/17ページ)

スピン (1/2 ページ)

スピン ボックスを使用すると、リストから値を選択できます。2 つのボタンを使用して、数値を増減したり、オプションのリストから項目を選択したりできます。

パネルに表示されるテキストは最初は "Value" なので、パネルを開いているときにテキストを初期化するコマンドを送信する場合があります。



使用例 : 室内の温度を適切に設定します。

コマンド

ボタン間に表示されるテキストを追加または更新するには、`SetValue`コマンドを使用します。

例 : "myspinner" の WdgetId を持つスピナーには、増加 (上矢印) ボタンと減少 (下矢印) ボタンの間にテキスト "22" を追加します。

端末モード

スピン ボックスの間でテキストを "22" に設定するには、次のコマンドを使用します。

```
xCommand UserInterface Extensions Widget SetValue WidgetId: "myspinner"
Value: "22"
```

マクロ

スピン ボックスの間でテキストを "22" に設定するには、次のコマンドを使用します。

```
xapi.command('UserInterface Extensions Widget SetValue', {WidgetId:
'myspinner', value: '22'});
```

ウィジェット (13/17ページ)

スピン (2/2 ページ)

イベント

UI で変更を監視します。スピン ボックスには、次のタイプのイベントが発行されます。

- *Pressed* - スピナーボタンのいずれかのボタンを押すとトリガーされます。
  値 : <increment/decrement>
- *Released* - スピナーボタンのいずれかのボタンを離すとトリガーされます。
  値 : <increment/decrement>
- *Clicked* - スピナーボタンのいずれかのボタンを離すとトリガーされます。
  値 : <increment/decrement>

例 : "myspinner" の WidgetId を持つスピン ボックスの減少 (下矢印) ボタンを押してから離します。端末およびマクロ経由で、これらを検出します。

端末モード

端末を介したウィジェットのイベントに登録します。例 :

```
xFeedback Register Event/UserInterface/Extensions/Widget/Action
```

ボタンを押して離すと、いくつかのイベントがブロードキャストされます。
例 :

```
*e UserInterface Extensions Widget Action WidgetId: "myspinner"
*e UserInterface Extensions Widget Action Value: "decrement"
*e UserInterface Extensions Widget Action Type: "pressed"
** end
*e UserInterface Extensions Widget Action WidgetId: "myspinner"
*e UserInterface Extensions Widget Action Value: "decrement"
*e UserInterface Extensions Widget Action Type: "released"
** end
*e UserInterface Extensions Widget Action WidgetId: "myspinner"
*e UserInterface Extensions Widget Action Value: "decrement"
*e UserInterface Extensions Widget Action Type: "clicked"
** end
```

XML モード

たとえば、XML は次のようになります。

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

マクロ

ユーザが上矢印および下矢印を押した時に表示される値を変更するには、次のコマンドを使用します。

```
let spinner_value = 22;
xapi.command('UserInterface Extensions Widget SetValue', {WidgetId: 'myspinner',
value: spinner_value});
xapi.event.on('UserInterface Extensions Widget Action', (event) => {
   console.log(event);
    if ((event.WidgetId === 'myspinner') && (event.Type === 'clicked')) {
        if (event.Value === 'increment') spinner_value++
        else spinner_value--
        xapi.command('UserInterface Extensions Widget SetValue', {WidgetId:
'myspinner', value: spinner_value});
    }
  });
```

ウィジェット (14/17ページ)

[テキスト (Text)]

テキストウィジェット は、ディスプレイ上にテキストを配置するために使用されます。ユーザは操作しません。

テキストのウィジェットのサイズは異なります。最大 2 行までのテキストを追加することができ、テキストは次の行に自動的に折り返されます。

フォント サイズが大きい、ライン ラップなしの小さなテキスト ウィジェットも使用できます。

エディタでテキスト ウィジェットに最初のテキストを指定し、後から `SetValue` コマンドを使用してテキストを動的に入力することができます。

使用例 : ヘルプ テキスト、操作の指示、各種のプリセットの説明、制御システムから通知される情報テキスト (「プロジェクターはウォームアップ中です」など) 。

フォント サイズが大きいテキスト ボックスは、主に現在の室温などのステータス値を示すために使用されます。

グループ内の個々のボタンが小さすぎて説明テキストを含めない場合、説明にテキストウィジェットを使用できます。

コマンド

テキスト ウィジェット内のテキストを変更するには、`SetValue` コマンドを使用します。

例 : "mytext": "The projector is warming up." があるてきすとウィジェットで次のテキストを設定します。

端末モード

端末を使用してテキストを設定するには、次のコマンドを使用します。

```
xCommand UserInterface Extensions Widget SetValue WidgetId: "mytext" Value: "The projector is warming up."
```

後で削除するには、スペースを使用します。

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

マクロ

マクロを使用してテキストを設定するには、次のコマンドを使用します。

```
xapi.command('UserInterface Extensions Widget SetValue', {WidgetId: 'mytext', value: 'The projector is warming up. '});
```

後で削除するには、スペースに設定します。それ以外の場合は "Text" と表示されます。

```
xapi.command('UserInterface Extensions Widget SetValue', {WidgetId: 'mytext', value: ' '});
```

イベント

テキスト ウィジェットにはイベントは関連付けされていませんし、インタラクティブではありません。

ウィジェット (15/17ページ)

方向パッド (1/2 ページ)

方向パッドは、5 つのボタン、つまり 4 つの方向ボタンと中央のボタンのセットと見なすことができます。

方向パッドは、デバイスの動きを 2 方向に制御するために使用されます。

使用例 : カメラまたは AppleTV の制御

コマンド

方向パッドに使用できるコマンドはありません。

ウィジェット (16/17ページ)

方向パッド (2/2 ページ)

イベント

UI で変更を監視します。方向性パッドに対して、以下の種類のイベントが発行されます。押されたボタンを示すために次のいずれかの値が提供されます。:
値は、「up」、「down」、「right」、「left」、「center」のいずれかです。

- *Pressed*いずれかのボタンを押すとトリガーされます。
  値 : <"up"/"down"/"right"/"left"/"center">
- *Released*いずれかのボタンを離すとトリガーされます。
  値 : <"up"/"down"/"right"/"left"/"center">
- *Clicked* - いずれかのボタンを離すとトリガーされます。
  値 : <"up"/"down"/"right"/"left"/"center">

例 : "mydirectionalpad" の WidgetId を持つ方向パッドのボタンを押して離します。端末およびマクロを介して、このアクションを検出します。

端末モード

端末を介したウィジェットのイベントに登録します。例 :

```
xFeedback Register Event/UserInterface/Extensions/Widget/Action
```

ボタンのいずれかを押して離すと、いくつかのイベントがブロードキャストされます。たとえば、「上」ボタンを押す場合は、次のようになります。

```
*e UserInterface Extensions Widget Action WidgetId: "mydirectionalpad"
*e UserInterface Extensions Widget Action Value: "up"
*e UserInterface Extensions Widget Action Type: "pressed"
** end
*e UserInterface Extensions Widget Action WidgetId: "mydirectionalpad"
*e UserInterface Extensions Widget Action Value: "up"
*e UserInterface Extensions Widget Action Type: "released"
** end
*e UserInterface Extensions Widget Action WidgetId: "mydirectionalpad"
*e UserInterface Extensions Widget Action Value: "up"
*e UserInterface Extensions Widget Action Type: "clicked"
** end
```

XML モード

たとえば、XML は次のようになります。

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

マクロ

ユーザが方向パッドの矢印を押した時に表示される値を変更するには、次のコマンドを使用します。

```
const xapi = require('xapi');
var x, x0, y, y0;
x = x0 = y = y0 = 50;

xapi.event.on('UserInterface Extensions Widget Action', (event) => {
    if ((event.WidgetId === 'mydirectionalpad') && (event.Type === 'clicked')) {
        switch(event.Value) {
          case 'right': x++; break;
          case 'left': x--; break;
          case 'up': y++; break;
          case 'down': y--; break;
          case 'center': // reset to original values
            x = x0;
            Y = y0;
            break;
        }
        console.log('x:', x, 'y:', y)
    }
});
```

ウィジェット (17/17ページ)

スペーサ

スペーサを使用すると、ウィジェット間、またはウィジェットの後にスペースを追加できます。

スペーサの幅は調節可能です (1 ～ 4) 。最大値に設定した場合、その行を占有します。また、垂直スペーサとしても使用できます。

スペーサはレイアウト ツールにすぎません。したがって、スペーサに関連付けられたイベントまたはコマンドはありません。

コマンド

スペーサに使用できるコマンドはありません。

イベント

スペーサにはイベントはありません。

UI 拡張機能のヒント (1/2ページ)

再起動後に登録

ビデオデバイスまたはコントロールシステムのいずれかが再起動した場合、コントロールシステムは、誰かがカスタム コントロールを使用するか新しいコントロール パネルをビデオデバイスにエクスポートしたときにビデオデバイスが送信するイベントに再登録する必要があります。

端末出力モードの場合：

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

XML 出力モードの場合：

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

詳細については、「アプリケーション プログラム インターフェイス (API)」の章を参照してください。

すべてのウィジェットを初期化する

次のような状況では、制御システムがパネル上のすべてのウィジェットを初期化していることを確認してください。

- 制御システムがビデオデバイスに最初に接続したとき。
- ビデオデバイスが再起動したとき。
- 制御システムが再起動したとき。
- 新しいコントロール パネルがビデオデバイスにエクスポートされたとき (`LayoutUpdated` イベントへの応答として)。

初期化を行わないと、システムが誤った値を表示し、室内の実際のステータスが反映されないことがあります。

初期値を設定するには、`SetValue` コマンドを使用します。

設定が変更されたら必ずビデオデバイスに値を戻します。

予期しない動作や曖昧さを回避するため、制御システムは、設定が変更されたら必ず `SetValue` コマンドをビデオデバイスに送信する必要があります。これは、システムのコントロールを使用しているユーザが変更をトリガーした場合にも適用されます。

たとえば、照明を暗くするためにパネルのスライダを使用しても、室内の物理的な調光器や別のタッチコントローラを使用しても違いはありません。制御システムは常に、`SetValue` コマンドを使用して調光器の値をビデオデバイスに送信する必要があります。

前の値の復元

以前の値を保存する。照明をオフにするときは (調光スライダとオン/オフのトグル ボタンが付いた照明など)、消灯時の状態を保存しておき、再度オンにするときにその値を使用します。

例：明るさが 40% のときに照明をオフにして、再度*オン*にすると、ユーザは照明が (100 % ではなく) 40% の明るさになると予想します。調光スライダの値が 0% になったときは、電源スイッチをオフに設定することも忘れないでください。

ウィジェット ID の使用

ウィジェット (テキスト フィールドなど) をページ上にドラッグするとき、カスタマイズされた ID を入れてください。ウィジェット ID を一意にする必要はありません。ウィジェット間で ID を共有できます。しかし、その場合は同じタイプのウィジェットである必要があります。つまり、異なるパネルで「メイン照明」という名前の 2 つのスライダを作成することはできますが、1 つをスライダ、他方をトグル ボタンにして両方に「メイン照明」と名前を付けることはできません。

別のページまたはパネルの既存のウィジェットの複製を作成するには、コピーして貼り付けます。

UI 拡張機能のヒント (2/2 ページ)

パネルの更新

新しいパネルをビデオデバイスにエクスポートすると、古いパネルが上書きされ、新しいパネルに置き換えられます。

アップデート:

1. ビデオデバイスの Web インターフェイスから **UI 拡張機能エディタ**を起動します。
2. 必要なコントロール パネルを作成するか、または以前に保存したパネルをファイルからインポートします (*Import > From file*) 。
3. *エクスポート (Export)] > [コーデックへ (To codec)]* をクリックします。

パネルの削除

ビデオデバイスにカスタムパネルがある場合は、ユーザインターフェイスに対応するボタンがあります。パネルが空で、ウィジェットを含まない場合でも、アイコンとパネルの両方が表示されます。

コントロールパネルとアイコンを削除するには、次の操作を実行します。

1. ビデオデバイスの Web インターフェイスから **UI 拡張機能エディタ**を起動します。
2. 削除するパネルを選択します。
3. [パネルの削除 (*Delete panel*)] をクリックします。

サード パーティのコントロールシステムから RoomOS への移行

サード パーティのコントロールシステムをすでに使用している場合、本書の説明に従って RoomOS の使用を開始するには、次のようにすることをお勧めします。

1. サードパーティ製の機器を制御するために作成したプログラミングコードをそのまま残します。
2. シスコのビデオデバイスを制御するすべてのコードを削除します。これは、すでに UI 拡張機能を介して制御されているためです。
3. サード パーティ制御システム パネルからのボタン押下によるシグナリングを再プログラムして、代わりにシスコのビデオデバイスからのボタン押下をリッスンするようにします。

最大のコントロールシステムメーカーが制御用のモジュール/ドライバを提供していますので、このプログラミングは非常に簡単です。

UI 拡張機能のトラブルシューティング

サインイン

ビデオデバイスの Web インターフェイスに管理者クレデンシャルでサインインし、*[統合 (Integration)]* > *[UI 拡張機能エディタ (UI Extensions Editor)]* に移動します。矢印をクリックして、[開発ツール (Development Tools)] を表示します。

すべてのウィジェットとそのステータスの概要

[ウィジェット状態の概要 (Widget State Overview)] ウィンドウには、すべてのウィジェットとそのステータスが一覧表示されます。ステータスは、[現在の値 (Current Value)] 列に表示されます。

[現在の値 (Current Value)] 列が空の場合、ウィジェットは初期化されておらず、値が設定されていません。コントロールシステムが最初にビデオ デバイスに接続するときに、すべてのウィジェットを初期化することをお勧めします。

更新された値のビデオデバイスへの送信

コントロールシステムは、`SetValue` コマンドをビデオデバイスに送信し、ウィジェットを更新するように指示します。テスト目的で、[ウィジェット状態の概要 (Widget State Overview)] ウィンドウの [値の更新 (Update Value)] 列を使用して、制御システムをシミュレートすることができます。

入力フィールドの 1 つに値を入力して、対応する `SetValue` コマンドをビデオデバイスにただちに送信します。*[現在の値 (Current Value)]* 列 (ステータス) が更新され、それに応じてタッチコントローラパネルが変更されます。

ウィジェットの値をクリアするには、*[設定解除 (Unset)]* をクリックします (`UnsetValue` コマンドを送信します) 。

コントロールシステムがビデオデバイスに接続されている場合、*[現在の値 (Current Value)]* 列と *[値の更新 (Update Value)]* 列が同期しなくなることがあります。[現在の値 (*Current Value*)] 列には常に現在のステータスが表示されます。SetValue コマンドが実際の制御システムや *[値の更新 (Update Value)]* 列から送信されているかどうかは関係ありません。

イベントとステータスの更新の確認

ウィジェットに関連するすべてのイベントとステータスの更新は、ただちに [ログ (Log)] ウィンドウに表示されます。イベントには、`*e` という接頭辞が付き、ステータスには `*s` という接頭辞が付きます。

ユーザインターフェイスのコントロールを使用すると、イベントが表示されます。ビデオデバイスのステータスを変更するコマンドをビデオデバイスに送信すると、ステータスが更新されます。

パネルのロードに失敗した場合

既存のパネルが **UI 拡張機能エディタ** の起動時に自動的にロードに失敗する場合は、ビデオデバイスからパネルを手動でインポートするか、バックアップ XML ファイルをロードする必要があります。

これらの選択肢によって、エディタ内の保存されていないデータは消去されますが、ビデオデバイスの既存のパネルは、新しいパネルがビデオデバイスにエクスポートされるまで上書きされたり、削除されたりすることはありません。

マクロを確認する

意図しない動作の変更が起こり、システムでマクロを実行する場合、トラブルシューティングを進める前に、必ずマクロを無効にします。

これを行うには、`xConfiguration Macros Mode: On/Off` を使用します。

マクロ フレームワークには、*macros.log* という名前の独自のログ ファイルがあります。

macros.log ファイルには、**コンソールログ**に出力されるものの多くが記録されます。コンソールに出力するようにマクロを設定できますが、これはログに保存されます。そのため、このファイルでカスタム ログ メッセージ (これは開発者が前もって作成する場合があります) を確認できることを覚えておいてください。

第 3 章

マクロ

マクロ フレームワーク

マクロは、Webex デバイスでローカルに実行される小さな javascript プログラムです。arrow functions、promises、classes、async/await などの標準の javascript 機能およびデバイスの状態を監視および制御できる xAPI コマンドを使用できます。

マクロフレームワークには多くの利点があり、インテグレータで以下を行えます。

- 導入の調整
- カスタム機能またはワークアラウンドの作成
- シナリオまたは再構成の自動化
- カスタム テストまたはモニタリングの作成

マクロと外部システムを組み合わせると便利です（プレゼンテーションのステータスや通話状態に応じて照明を調整するなど）。

ユーザインタフェイスに表示されるカスタマイズしたテキストをマクロに含めることができます。このテキストは、特定の機能のアクティブ化/非アクティブ化を確認するようユーザーに警告したり、メッセージに従って行動するようユーザーに注意を喚起したりするために使用できます。

このようなテキストによって純粋に情報を提供することができますが、情報を入力して応答するようユーザに求めることもできます。また、この情報に基づいて、ビデオデバイスに直接動作させることもできます。

マクロを使用する場合、UI 拡張機能でローカル機能を有効にするための外部コントロールシステムは不要です。

ただし、照明やブラインドなどの周辺機器を制御するような、API 経由でローカルアクションを実行する場合は、適切なサードパーティ製のコントロールシステムが必要になります。

ローカル機能の例には、短縮ダイヤル用のパネルや、すべての設定をデフォルトに戻す「ルーム リセット」がトリガーされる場合があります。

大量のマクロを使用すると、重い負荷がかかりビデオデバイスのパフォーマンスが低下する可能性があることに注意してください。

マクロは、システムの期待どおりの動作を変更する場合があります。マクロが通常のユーザを驚かせたり混同させる可能性がある場合は、ビデオデバイスの画面で通知を行うなどして、ユーザに知らせて下さい。

マクロを他のマクロに依存させないようにすることをお勧めします。もちろん、複数のマクロのアクションが相互に依存しない限り、呼び出し状態など同じ xapi 値を監視する複数のマクロを作成することはできます。

免責事項：シスコはマクロフレームワーク自体のみをサポートします。シスコでは、コンパイルに失敗するコード、または動作しないことが開発者の「意図」どおりであるコードはサポートしません。構文が正しいこと、また JavaScript でマクロを記述するための十分なコーディング スキルの取得は、コードを記述する人の責任になります。公開している開発者フォーラムは、マクロの作成を支援できます。

マクロ エディタの紹介

管理者のログイン情報を使用してビデオデバイスの Web インターフェイスにサインインし、*[カスタマイズ (Customization) > マクロ エディタ]* に移動します。

初回実行時、このビデオデバイスでマクロの使用を有効にするかどうか尋ねられます。

新しいマクロ プログラミング セッションを開始するには、*[ファイルからインポート (Import from file....)]* をクリックして、ファイル (*.js) から既存のコードをインポートします。または *[新しいマクロの作成 (Create new macro)]* をクリックして空白のマクロを作成します 。

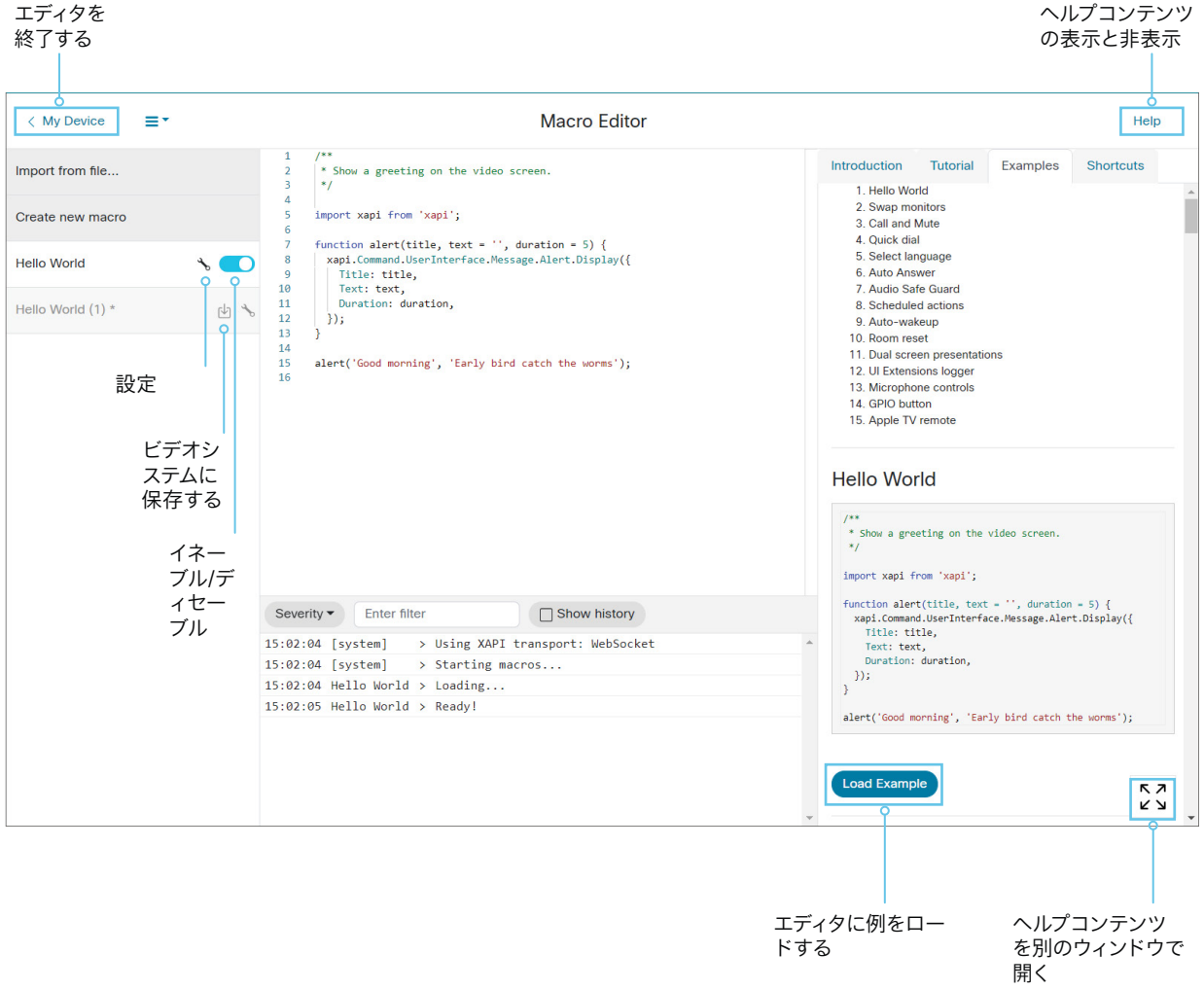
たとえば、*[ヘルプ]* セクション > *[例]* のタブから「Hello World」の例を探します。*[例のロード (Load Example)]* をクリックします。

エディタに新しいマクロの例が表示されます。新しいマクロの *[保存 (Save)]* アイコンをクリックすると、*[無効化/有効化]* トグルが表示されます。トグルしてマクロを有効にします。

この場合、マクロが自動的に実行され、タッチコントローラまたは画面にグリーティングが表示されます。

マクロエディタには 、例やチュートリアルが多数含まれています。開始とコマンド専用のウィジェットの例については、「ウィジェット (Widgets)」セクションを参照してください。

Web 上で使用できる便利な方法や例については、「オンラインリソース」を参照してください。



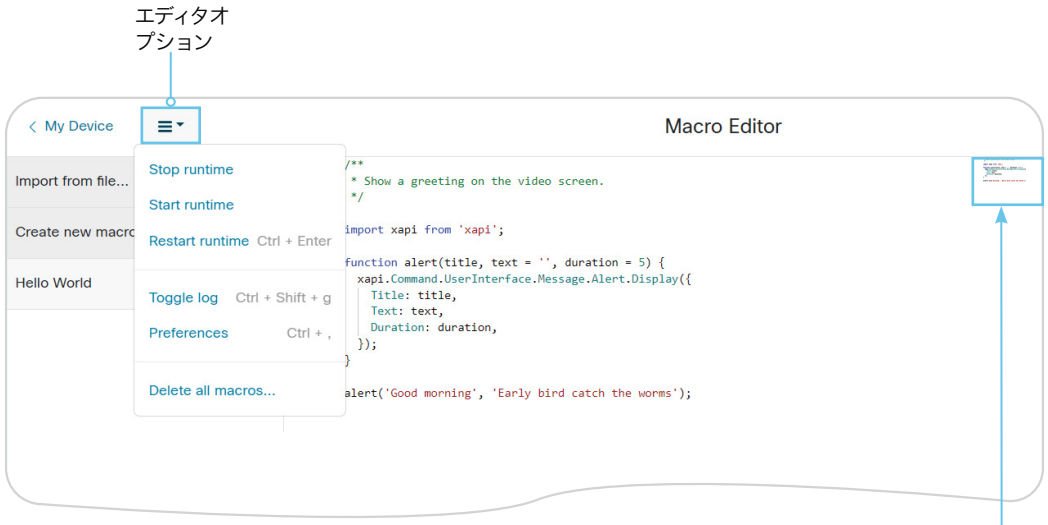
マクロ エディタの紹介 (2/3 ページ)

マクロの保存、削除、またはアクティブ/非アクティブの切り替えを行うと、ランタイム全体が再起動され、すべてのマクロが再実行されます。トラブルシューティング中に未使用のマクロをすべて無効にします。

マクロ エディタは自動的に構文エラーを検出し、スクリプトに検出されたエラーがある場合、コードを保存しないようにします。エラーの上にカーソルを動かし、詳細を確認します。

ドロップダウン リストのオプションを使用して、実行時にランタイムを起動、停止、または再起動できます。

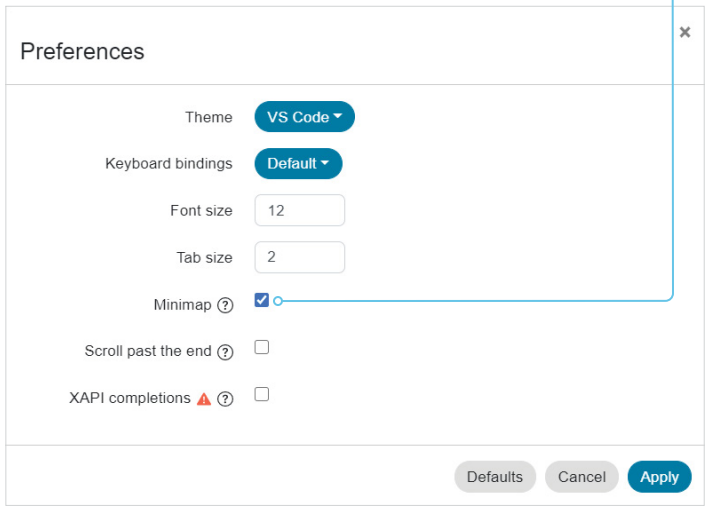
- *停止* - ランタイムを停止します。これにより、すべてのマクロが停止されます。
- *起動* - 実行時にランタイムを起動します。これにより、有効なマクロがすべて自動的に実行されます。
- *再起動* - ランタイムを停止して再起動します。これにより、有効なマクロがすべて自動的に実行されます。
- *ログの切り替え* - **ログ コンソール** のペインを非表示または表示します。
- *設定* - *設定*ダイヤログボックスを開きます。
- *すべてのマクロを削除* - エディタからすべてのマクロを削除します。



*設定*ボックスには、**マクロ エディタ**に関する複数の プションがあります。

- *テーマ* - エディタ用の色のテーマを選択します。
- *キーボードバインド* - 使用するキーボードのショートカットセットを定義します。オプションは、*Default*、*Vim* と *Emacs* です。
- *フォントサイズ* - 文字のフォントサイズを定義します。
- *タブサイズ* - エディタが各タブのインデントに使用するスペースの数を定義します。
- *ミニマップ* - ミニマップの概要スクロールの表示を切り替えます。スクリプトの小さなマップがエディタの右側に表示され、完全なスクリプト内での位置をお知らせします。長いスクリプトに最も役立ちます。
- *最後までスクロールする* - マクロの末尾が画面の中央で維持されます。
- *XAPI の完了* - 自動単語補完編集を有効にして、エディタが入力している残りの単語を予測できるようにします。

  自動完了が無効な場合は、*Ctrl-Space* を使用して候補を表示することもできます。



マクロ エディタの紹介 (3/3 ページ)

ログ コンソールの目的は、マクロを実行すると何が起こるかを明らかにすることです。

ウィンドウの下部に**ログコンソール**が表示されない場合は、そのログコンソールを *[Runtime（ランタイム）]* > *[ログの切り替え（Toggle Log）]* メニューまたは `Ctrl+Shift+G` をタイプして表示できます。

ログコンソールは 、ランタイムとマクロのアクションについてレポートします。マクロの実行中、情報をコンソールに出力することもできます。

たとえば、新しいマクロを作成し、次の行を追加します。

```
console.log('Hello World!');
```

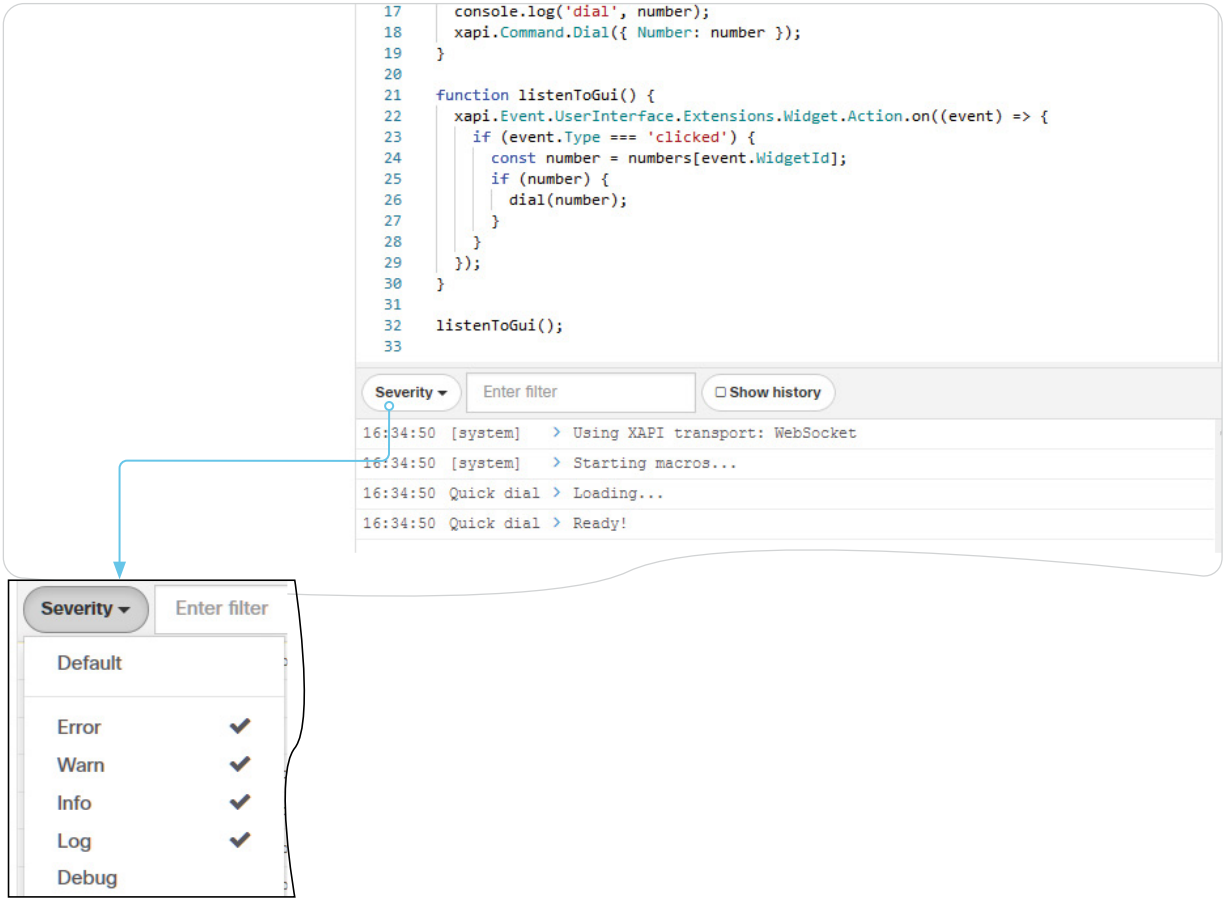
`Ctrl+S` を押して、マクロを保存して実行します。**ログ コンソール** を確認すると、ログにメッセージが表示されます。

コンソールメッセージなどのエラーメッセージは、マクロの問題について理解を深める際に役立つ情報です。

使用可能なオプションは複数あります。

- *シビラティ（重大度）* - ログに表示する情報のレベルを選択します。オプションには、*デフォルト* 、*エラー*、*警告*、*情報*、*ログ*、*デバッグが含まれます。*
  ログ記録する情報のタイプがここでチェックされます。それ以外の場合は、コンソール ログのウィンドウに表示されません。
- *フィルタの入力* - ログのフィルタに使用する必要があるテキストを入力します。入力したテキストを含むメッセージだけが表示されます。
- *履歴の表示* - 有効にすると、ログメッセージの履歴全体が表示されます。それ以外の場合は、前回の再起動以降に発生した情報だけを表示します。

ログ コンソールペインに表示される情報のほとんどは、ビデオデバイスの *macros.log* ファイルに書き込まれます。



マクロ ランタイム

ランタイムについて

すべてのアクティブなマクロは、ビデオデバイスの単一プロセスで実行されます。これを *ランタイム*と呼んでいます。ランタイムはサンドボックス化されているため、実行されるコードはデバイスの標準ソフトウェアから安全に分離されます。

タンタイムは、デフォルトで実行されますが、**マクロエディタ**から手動で停止/開始できます。デバイスを再起動して、`xconfiguration macros autostart` を *On* にすると、ランタイムが自動で再起動します。

マクロが応答しなくなると（無限ループなどのために数秒間応答できなくなることがあります）、安全メカニズムによってがランタイム停止され、それにより、すべてのマクロも停止します。

数秒後、ランタイムは自動で再起動されます。この動作は継続しますが、ランタイムがシャット ダウンされるたびに、再起動までの時間が長くなります。これが一定回数以上発生すると、マクロに問題があることを通知するためにシステム診断が表示されます。

マクロの有効化/無効化

各マクロを**マクロエディタ**から無効にすることもできます。

ランライムの自動再起動を無効化するには、`xConfiguration Macros Mode: Off` を使用する必要があります。

システムによる予期しない動作が発生する場合に、このコマンドを使用することができます。このような場合、トラブルシューティングをする前にマクロを常に無効化する必要があります。

トラブルシューティング

意図しない動作の変更が起こり、システムでマクロを実行する場合、トラブルシューティングを進める前に、必ずマクロを無効にします。

これを行うには、`xConfiguration Macros Mode: On/Off` を使用します。

マクロ フレームワークには、*macros.log* という名前の独自のログ ファイルがあります。

macros.log ファイルには、**ログ コンソール**に出力されるものの多くが記録されます。コンソールに出力するようにマクロを設定できますが、これはログに保存されます。そのため、このファイルでカスタム ログ メッセージ（これは開発者が前もって作成することが必要です）を確認できることを覚えておいてください。

関連資料

マクロ作成用の教材は、**マクロエディタ**のオンラインヘルプセクションを参照してください。

Web 上で使用できるチュートリアルや例へのリンクについては、「オンラインリソース」項を参照してください。

第 4 章

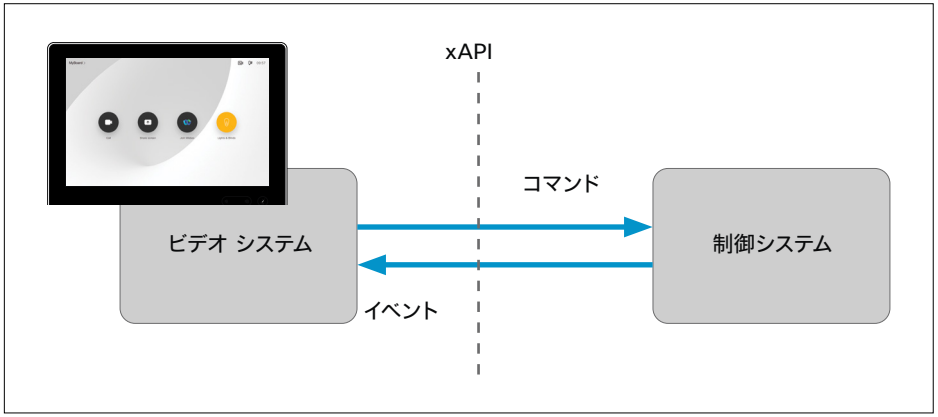
アプリケーション プログラム インターフェイス (API)

xAPI を介した通信

ビデオデバイスとコントロールシステムは、xAPI を介してメッセージを交換し、ユーザインターフェイスが常にルームの実際のステータスを反映するようにします。

カスタム ユーザインターフェイス パネルのコントロールのいずれかを使用している場合、ビデオデバイスは 1 つ以上のイベントを送信します。ルーム設定に変更がある場合は、コントロールシステムからビデオデバイスにコマンドを送信する必要があります。

ユーザーインターフェイスの [照明オン (Lights On)] ボタンをタップすると 、ビデオデバイスが関連イベントを送信します。コントロールシステムは、これらのイベントに応答するために室内の照明をオンにして、対応するコマンドをビデオデバイスに送り返す必要があります。



ビデオ システムと制御システムは、xAPI を介してメッセージを交換します。

ビデオデバイスへの接続

xAPI により、ビデオデバイスと AMX や Crestron などのサードパーティのコントロールシステムとの双方向通信を可能にします。次のような複数の方法で xAPI にアクセスできます。

- Telnet
- SSH
- HTTP/HTTPS
- RS-232/シリアル接続

どの方法を選択しても、xAPI の構造は同じです。使用するアプリケーションおよびビデオデバイスに最適なアクセス方式を選択してください。

使用可能なアクセス方式と xAPI の使用方法の詳細については、使用するビデオデバイスの API ガイドを参照してください。

次のリンクからアクセスできます。

- ▶ Cisco Webex Board シリーズ
- ▶ Cisco Webex Desk シリーズ
- ▶ Cisco Webex Room シリーズ

その後、*[リファレンス ガイド (Reference Guides)] > [コマンド リファレンス (Command References)]* をクリックして API ガイドを見つけます。

ビデオデバイスに接続されている周辺機器として、コントロールシステムを登録できます。

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

ここの `ID` とは、コントロールシステムの固有の識別子です。これは通常、MAC アドレスです。

ハートビート。コントロールシステムは、ビデオデバイスがまだ接続されていることを通知するために、次の信号を送信する必要があります。

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

ここで、`ID` は 制御システムに固有の ID です（通常は MAC アドレス）。**タイムアウト**は、各ハートビート間の秒数です。`Timeout` が指定されていない場合は、60 秒に設定されます。

ビデオデバイスが調整を受ける限り、コントロールシステムは接続されているデバイスのリストに残ります。ご使用のデバイスの API ガイドの「`xStatus Peripherals ConnectedDevice`」を参照してください。

接続されたユニットがハートビートの送信を停止した場合、ビデオデバイスがハートビートの停止を検出するまでしばらく時間が経過します。これには 2 分かかる場合があります。

同様に、ビデオデバイスがコントロールシステムから新しい停止を検出するまでに、数分が経過する場合があります。

UserInterface Extensions コマンドの使用

ウィジェットの値を設定する `SetValue` コマンドは、UI 拡張機能を使用する際に必要不可欠です。

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

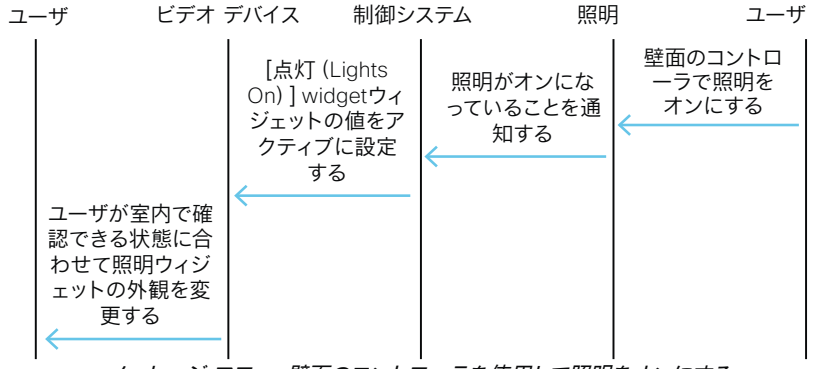
ビデオデバイスが `SetValue` コマンドを受け取った場合は、ビデオデバイスのステータスとユーザインターフェイスのカスタムパネル が必要に応じて更新されます。

次のような状況では、コントロールシステムが `SetValue` コマンドを送信することが重要になります。これにより、ビデオデバイスが室内の実際の状態を反映するようになります。

- コントロールシステムがビデオデバイスに最初に接続したとき。
- ビデオデバイスが再起動したとき。
- 制御システムが再起動したとき。
- 新しいカスタムパネルが**ユーザインターフェイス拡張機能エディタ**からビデオデバイスにエクスポートされたとき (LayoutUpdated イベントへの応答)。詳細については、「ユーザーインターフェイス拡張機能」を参照してください。
- 室内にある制御対象の機器の状態が物理的に変更されたとき (壁面のコントローラを使用して照明をオンにするなど)。
- イベントへの応答として (カスタムパネルの *[点灯 (Lights On)]* ボタンをタップした場合など)。
- コントロールシステムは、カスタム UI パネル上のアクションを反映するためにルーム内で必要なすべての操作を行う必要があります (物理的に照明をオンに切り替えるなど)。

例

メッセージフロー : カスタムパネルのコントロールを使用して照明をオンにします。



メッセージ フロー : 壁面のコントローラを使用して照明をオンにする

コマンド リファレンス

UserInterface Extensions Widget SetValue

このコマンドは、指定されたウィジェットの値を設定します。それに応じて UserInterface Extensions のステータスが更新されます。範囲外の値を設定しようとすると、エラーが返されます。

使用方法:

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

引数:

値: 文字列 (0、255) - ウィジェットの値。値の範囲は、ウィジェット タイプによって異なります。

WidgetId: 文字列 (0、40) - ウィジェットの一意の識別子。

UserInterface Extensions Widget UnsetValue

このコマンドは、指定されたウィジェットの値を空にします。それに応じて UserInterface Extensions のステータスが更新されます。そのウィジェットの選択が解除されたことがユーザインターフェイスに通知されます。

使用方法:

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

引数:

WidgetId: 文字列 (0、40) - ウィジェットの一意の識別子。

UserInterface Extensions Clear

このコマンドは、ビデオデバイスからすべてのユーザインターフェイス拡張 (ウィジェット) を削除します。

使用方法:

```
xCommand UserInterface Extensions Clear
```

UserInterface Extensions List

このコマンドを使用して、ビデオデバイスに存在するすべてのユーザインターフェイス拡張 (ウィジェット) を一覧表示します。

使用方法:

```
xCommand UserInterface Extensions List
```

参照先のステータス

UserInterface Extensions Widget [n] WidgetId

UserInterface Extensions Widget [n] Value

このステータスは、ウィジェットの識別子 (WidgetId) と現在の値を返します。

`UserInterface Extensions Widget SetValue` コマンドを使用して値が設定されるまで、値は空の文字列です。

使用方法：

```
xStatus UserInterface Extensions
```

返される結果の値スペース：

値：：ウィジェットの値。ウィジェット タイプによって異なります。文字列 (0、255)

WidgetId：ウィジェットの固有識別子。文字列 (0、40) 。

例：

```
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
```

イベント参考資料 (1/4ページ)

ビデオデバイスは、ユーザインターフェイスのカスタムパネルのコントロールを使用している場合に、次のイベントの 1 つ以上を送信します。

- **Pressed**：ウィジェットを最初に押したときに送信されます。
- **Changed**－ウィジェットの値を変更したときに送信されます (トグルボタンとスライダのみに適用されます)
- **Released**：ウィジェットから指を離したときに送信されます (ウィジェット外に指を移動してから離したときも送信されます)。
- **Clicked**：ウィジェットをクリックしたときに送信されます (ウィジェット外に指を移動しないで押して離した場合)。

これらのイベントは、次の 2 つの方式で送信されます。

- **UserInterface Extensions Event**：端末出力モード向け
- **UserInterface Extensions Widget**：XML 出力モード向け

登録する制御システムに最適な方式を確認するには、右の表を参照してください。

これらのイベントがトリガーされるウィジェットと、そのタイミングは、「ウィジェット (Widgets)」セクションに説明されています。

UserInterface Extensions Event (端末出力モード)	**UserInterface Extensions Widget** (XML 出力モード)
単一の文字列に、アクションのタイプ、イベントをトリガーしたウィジェット (ウィジェット ID で識別されます)、およびウィジェットの値に関する情報が含まれます。	アクションのタイプ、イベントをトリガーしたウィジェット (ウィジェット ID で識別されます)、およびウィジェットの値は、XML ツリー内の別々の要素として含まれます。

登録方法：
`xfeedback register event/UserInterface/Extensions/Event`
 	
例：
`*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`
	登録方法：
	`xfeedback register event/UserInterface/Extensions/Widget`
	
	例：
	`<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>`

制御システムが登録できる 2 つのイベント方式：それぞれ、端末出力モードと XML 出力モードに適しています。

イベント参照先 (2/4 ページ)

パネル更新のイベント例

新しいコントロールパネルが適用されると、ビデオデバイスは次のイベントを送信します。

LayoutUpdated：新しいパネルをビデオデバイスにエクスポートするときに送信されます。

このイベントへの応答として、制御システムは、すべてのウィジェットを初期化するコマンドを送信して、室内設定の実際の状態を反映させる必要があります。

登録方法：

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

例：

端末出力モード：

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

XML 出力モード：

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

ページを開くまたは閉じるイベントの例

それぞれのページに固有のページ ID を与えた場合は、ページの開/閉時にシステムがイベントを送信できます。

EventPageOpened：ページが開かれたときに送信されます

EventPageClosed：ページが閉じたときに送信されます

ページの動作はラジオ ボタンと似ていて、別のページを開くと現在のページが閉じます。その場合は EventPageClosed と EventPageOpened の両方が発行されます。

登録方法：

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

例：

端末出力モード：

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

XML 出力モード：

```
<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>
```

PageClosed の例は、例の `Opened` を単に `Closed` に置き換えたものになります。このイベントを使用する標準的な方法は、イベントに基づきコントローラに何らかのアクションを実行させることです（この場合は AppleTV ボックスを自動的にオン/オフにする）。

イベント参照先 (3/4 ページ)

UserInterface Extensions Event Pressed

ウィジェットを最初に押したときにビデオデバイスによって送信されます。

"Pressed" タイプの UserInterface Extensions Widget Action イベントに相当します。

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

引数:

Signal: 文字列 (0, 255) - 文字列の形式は "<WidgetId>:<Value>" です。ここで、<WidgetId> はイベントをトリガーするウィジェットの固有識別子で、<Value> はウィジェットの値です。有効な値の範囲は、ウィジェット タイプによって異なります。

UserInterface Extensions Event Changed

ウィジェットの値を変更するときにビデオデバイスから送信されます。これはトグルボタンとスライダーにのみ適用されます。

"Changed" タイプの UserInterface Extensions Widget Action イベントに相当します。

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

引数:

Signal: 文字列 (0, 255) - 文字列の形式は "<WidgetId>:<Value>" です。ここで、<WidgetId> はイベントをトリガーするウィジェットの固有識別子で、<Value> はウィジェットの値です。有効な値の範囲は、ウィジェット タイプによって異なります。

UserInterface Extensions Event Released

ウィジェットから指を離したときにビデオデバイスによって送信されます (ウィジェット外に指を移動してから離しても有効になります)。

"Released" タイプの UserInterface Extensions Widget Action イベントに相当します。

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

引数:

Signal: 文字列 (0, 255) - 文字列の形式は "<WidgetId>:<Value>" です。ここで、<WidgetId> はイベントをトリガーするウィジェットの固有識別子で、<Value> はウィジェットの値です。有効な値の範囲は、ウィジェット タイプによって異なります。

UserInterface Extensions Event Clicked

ウィジェットをクリックしたときにビデオデバイスによって送信されます (ウィジェット外に指を移動しないで押して離した場合)。

"Clicked" タイプの UserInterface Extensions Widget Action イベントに相当します。

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

引数:

Signal: 文字列 (0, 255) - 文字列の形式は "<WidgetId>:<Value>" です。ここで、<WidgetId> はイベントをトリガーするウィジェットの固有識別子で、<Value> はウィジェットの値です。有効な値の範囲は、ウィジェット タイプによって異なります。

イベント参照先 (4/4 ページ)

UserInterface Extensions Widget LayoutUpdated

UI 拡張機能エディタからビデオデバイスに新しい設定をエクスポートする場合など、ユーザインターフェイス拡張機能の設定ファイルが更新された場合にビデオデバイスから送信されます。

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

XML 内:

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

UserInterface Extensions Widget Action

誰かがユーザインターフェイスのコントロールのいずれかを使用している場合、ビデオデバイスから送信されます。`UserInterface Extensions Event Type` イベントに相当します。

アクション タイプに応じて異なりますが、このイベントは次のイベントのいずれかに相当します。

```
*e UserInterface Extensions Event Pressed
*e UserInterface Extensions Event Changed
*e UserInterface Extensions Event Released
*e UserInterface Extensions Event Clicked Events
```

XML 内:

```
<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>
```

引数:

WidgetId: 文字列 (0、40) イベントをトリガーしたウィジェットの一意の識別子。

値: 文字列 (0、255) - ウィジェットの値。有効な値の範囲は、ウィジェット タイプによって異なります。

Type: <Pressed/Changed/Released/Clicked>

- **Pressed**: ウィジェットを最初に押したときに送信されます。
- **Changed**: ウィジェットの値を変更したときに送信されます (トグル ボタンとスライダのみ)。
- **Released**: ウィジェットから指を離したときに送信されます (ウィジェット外に指を移動してから離しても送信されます)。
- **Clicked**: ウィジェットをクリックしたとき (ウィジェットを押してウィジェット外に指を移動しないで離したとき) に送信されます。

第 5 章

オーディオ コンソール

オーディオ接続をカスタマイズする

オーディオコンソールは、Codec Pro を使用するシステムで利用できます。また、ビデオデバイスの Web インターフェイスの *[カスタマイズ (Customization)]* で確認できます。

このユーティリティを使用すると、簡単なドラッグ アンド ドロップを使用して、オーディオの入力と出力をどのように接続するかを定義できます。

論理入力グループと論理出力グループを定義することから始めます。これらには、ユーザが定義した名前を付けることができます。その論理入力と論理出力に物理入力と物理出力を割り当てます。

新しい論理グループは、いつでも追加または削除できます。

設定に適用された変更は直ちに保存され、有効になります。

物理出力は、1 つ以上の論理出力グループに割り当てできません。ただし、マイクなどの物理入力を複数の入力に割り当てることができます。これは、ローカル聴衆者がマイクによる話を聞く必要がある講堂でビデオデバイスを使用するなど、ローカル補正を操作するのに便利です。

オーディオ コンソールのセットアップでは、遠端に送信されるマイク信号の一部に*エコーコントロール*を使用できます。同時に、ローカルで使用される部分については省略しています。これは、*Direct: On* を使用して、そのマイクを複数の論理グループに割り当てる方法で行われます。

マイク信号にノイズ リダクションとイコライザの設定を適用することもできます。

Codec Pro には専用回線入力がありません。1 つ以上の回線入力は、ファンタム電源を無効にして入力ゲイン (レベル) を下げることにより、回線レベル入力に使用できます。

ただし、これらの入力がプレゼンテーションの音声配信に使用される場合、この方法では、マイクのミュートボタンを押したときに入力がミュートされたままになるという制限があります。

別のデバイスに関する詳細については、「*追加スピーカーおよび音の強化ガイドの使用*」を参照してください。次のリンクからアクセスできます。

https://www.cisco.com/go/in-room-control-docs

Web からのオーディオ

WebRTC コールまたは他の WebEngine 関連のソースからオーディオを受信するには、オーディオコンソールに、スピーカーとともに出力グループにルーティングされる WebView 1 コネクタを含む入力グループが必要です。

最初にオーディオコンソールを設定したときに WebEngine モードが有効になっている場合、この入力グループはすでに配置されています。ただし、最初の設定後に WebEngine モードが有効にした場合は、グループを手動で作成する必要があります。

オーディオ リターン チャネルの使用

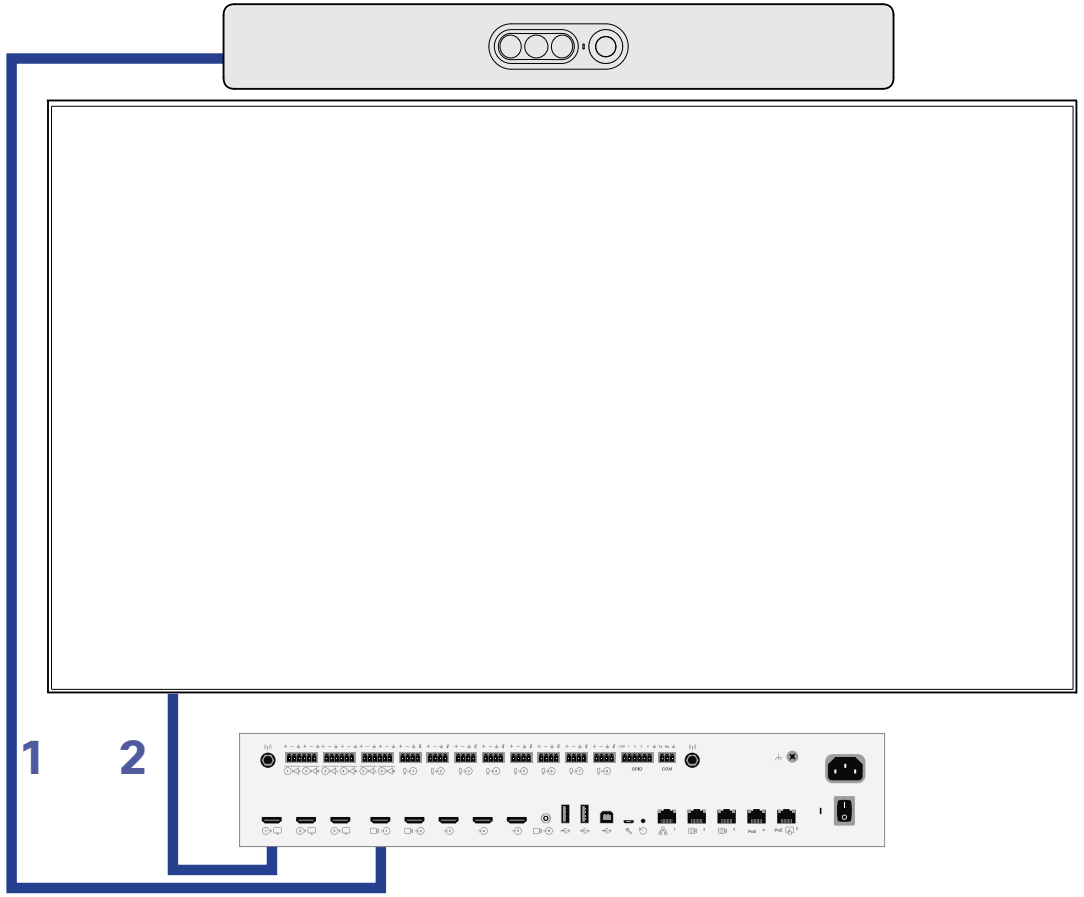
HDMI は、特定の環境下で、音声をどちらの方向にも送信できる能力を備えています。音声が逆方向に送信される場合をオーディオ リターン チャネル (ARC) と言います。Codec Pro は、ARC の使用をサポートしています。

図示のような設定を考えてみます。HDMI 経由で接続された Cisco QuadCam (一番上)、画面 (真ん中)、Codec Pro (一番下) が示されています。この設定では、QuadCam がカメラとサウンドバーの両方として機能します。

通常の用途では、1 の HDMI が QuadCam ユニットのカメラからの映像を Codec Pro に提供するために使用され、同じ HDMI のオーディオ リターン チャネルが音声を Codec Pro から QuadCam のスピーカーにルーティングするために使用されます。

また、セットアップを単なるサウンドバー付きの TV として使用する場合は、システムが Codec Pro の HDMI out 1(ARC) 経由で音声をモニタから Codec Pro に送信し、Codec Pro が HDMI1 in (ARC) 経由でその音声をさらに QuadCam に送信します。

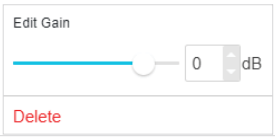
これを可能にするには、モニタを CEC+ARC 対応にする必要があります。セットアップで 4k 映像を使用する場合は、モニタが 4k 形式の CEC+ARC をサポートすることを確認してください。



オーディオ コンソール パネル

Codec Pro を使用するシステムの場合、**オーディオ コンソール**は、デバイスの Web インターフェイスの *[カスタマイズ (Customization)]* で確認できます。

接続の削除、またはゲインの調整を行うには、ケーブルのイラストの上で右クリックします。



デフォルトにリセットすると、検出された接続デバイスに基づいて一定の接続が作成されます。

物理入力コネクタのプールが使用可能です。

物理入力コネクタは論理入力グループに割り当てられます。これらの論理入力グループはユーザが作成します。

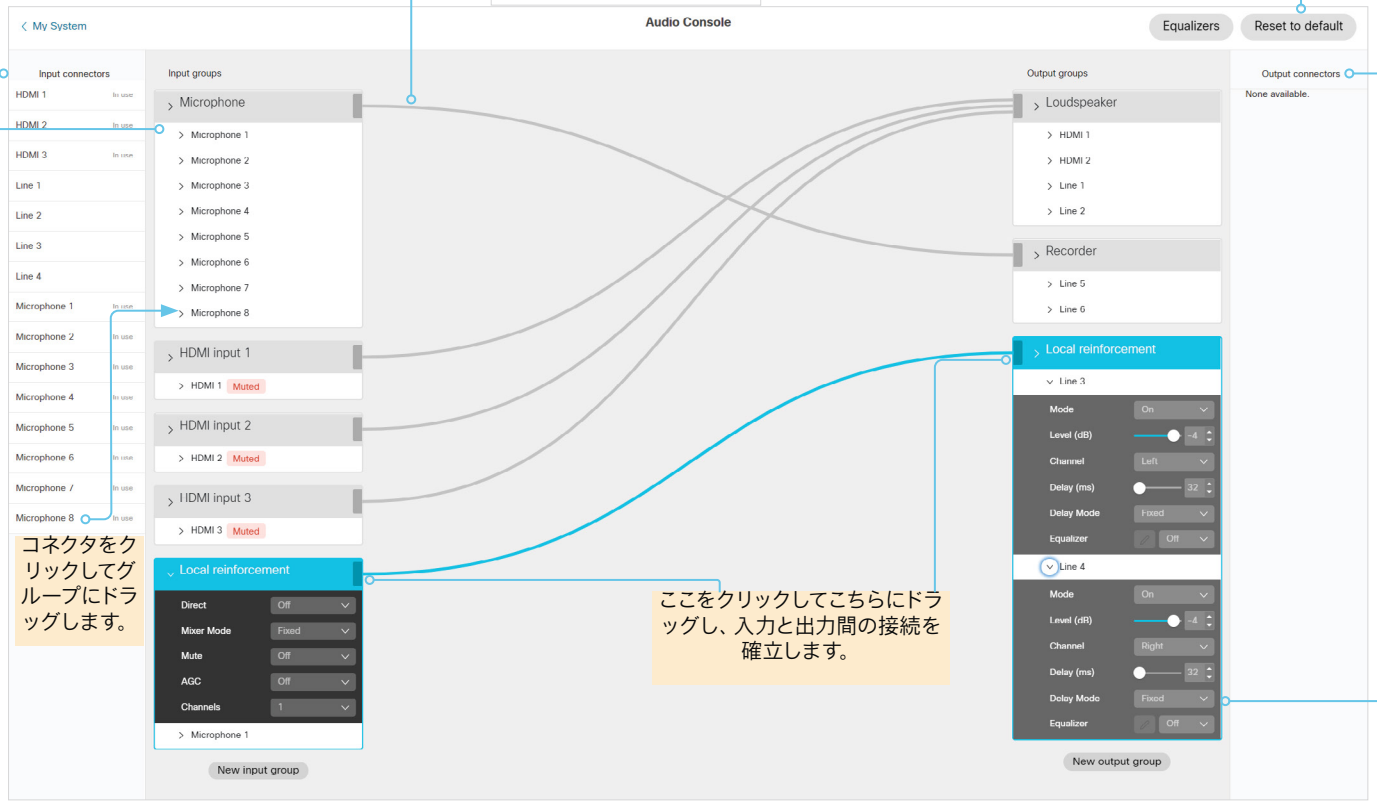
論理グループの名前は、*[新規入力グループ]* をクリックして指定します。

特定の物理入力コネクタを複数の論理グループに割り当てることもできます。ドラッグ アンド ドロップを使用します。

グループまたはグループのメンバーを削除するには、グループまたはメンバーにマウスをホバーします。X が表示されます。これをクリックすると、そのアイテムが削除されます。

ローカル補正に使用する場合は、個々のメンバーではなく論理グループを拡張します。

ローカル補正のシナリオでは、マイクを*[Direct (ダイレクト)]* に設定して、エコー制御などの余分な処理をバイパスします。



コネクタをクリックしてグループにドラッグします。

ここをクリックしてこちらにドラッグし、入力と出力間の接続を確立します。

これは、物理出力コネクタのプールです。使用されていないものは、「非アクティブ」と見なされます。

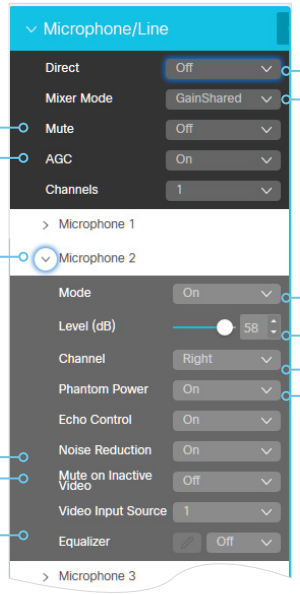
論理出力グループの処理は論理入力グループと同様ですが、物理出力を複数の論理出力グループに割り当てることはできません。

Codec Pro では追加でオーディオ リターン チャンネル (ARC) を選択できます。

外部画面によって発生する可能性のある遅延を回避するように*遅延モード*を*固定*することを推奨します。

これにより、遅延が最小限に抑えることができ、メイン音量コントロールがローカルプレゼンタの音量に影響を与えることを回避するためにも機能します。

マイクのセットアップについての詳細



論理グループ全体をミュートします。

論理グループ全体に適用されるAgc (自動ゲイン制御) を有効または無効にします。

チャネルには、1 (モノラル) または 2 (ステレオ) を使用できます。Codec Plus にのみ適用されます。

論理グループを展開して、グループレベルの設定にアクセスすることができます。

ノイズリダクション:
これを *[オン (On)]* に設定すると、室内に存在する連続雑音 (ファンの雑音など) が減少します。Impulsive ノイズは減少しません。

非アクティブなビデオのミュート: プレゼンターのマイクなど、シナリオではこの機能を使用します。このようにカメラ/マイクの組み合わせが設定されている場合、このマイクは、関連付けられたカメラ (プレゼンターを撮影しているもの) がビデオを実際に送信しない限り、ミュートされます。

イコライザ: イコライザでは、定義済みのイコライザ設定を最大 8 つ (またはなし、*Off* を示す) の中 から選択できます。

イコライザをセットアップするツールについては、イコライザーの設定にある説明を参照してください。

ローカル補正のシナリオでは、*エコー制御*などの余分な処理がバイパスされるように、[マイク (Microphone(s))] を *[ダイレクト (Direct)]* から *オン*に設定されていることを確認してください。これにより、遅延が最小化され、マスター音量コントロールがローカルプレゼンターの音量にも影響を与えることを回避するのに機能します。その他の使用例では、そのままにしておきます。

*ミキサーモード*は、 *GainShared*または*Fixed*にすることができます。*GainShared* が選択されている場合、マイクの信号の合計レベルが特定の値を超えることはありません (ただし、個々のオフセットは保持されます) 。*固定*モードでは、レベルの合計がまとめられます。

モード: マイクは*[オン (On)]*または*[オフ (Off)]*に設定できます。

レベル (dB) は 、ここで*ゲイン*と解釈する必要があります。

*チャンネル*が 2 (グループレベル) に設定されている場合は、下位レベルでチャンネル設定を使用して、この特定のマイクが左チャンネルのマイクグループに属するか、または右チャンネルのマイクグループに属するかを指定できます。

Codec Pro は、任意のタイプのマイクを含む入力ソースを使用してこれを実行できます。

論理グループ全体がグループレベルで *ダイレクト*に設定されている場合、*エコー制御*はN/Aに設定されます。

ファントム電源:
この設定では、マイクケーブルから電源を送信します。これは、回線レベルの信号を提供するプリアンプでマイクを使用している場合にオフにできます。

回線入力として Mic 入力を使用する場合は、ゲイン (通常は 40 dB 以上) を削減してください。

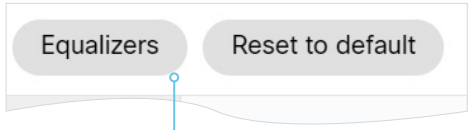
Codec Pro では、マイク入力は*ファントム供給*がアクティブになっているライン入力です。

イコライザーの設定

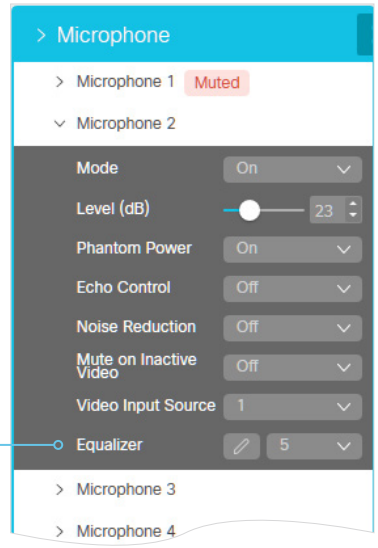
これには 8 つのユーザー定義可能なパラメータ化された均等化設定があります。

設定は、1 つのフィルタタイプ、ゲイン、中央、クロスオーバー周波数、および Q 値を持つ最大 6 つのセクションで構成されています。

各セクションには独自の色が表示されます。白色の線は、イコライザの結合された周波数応答を示しています。



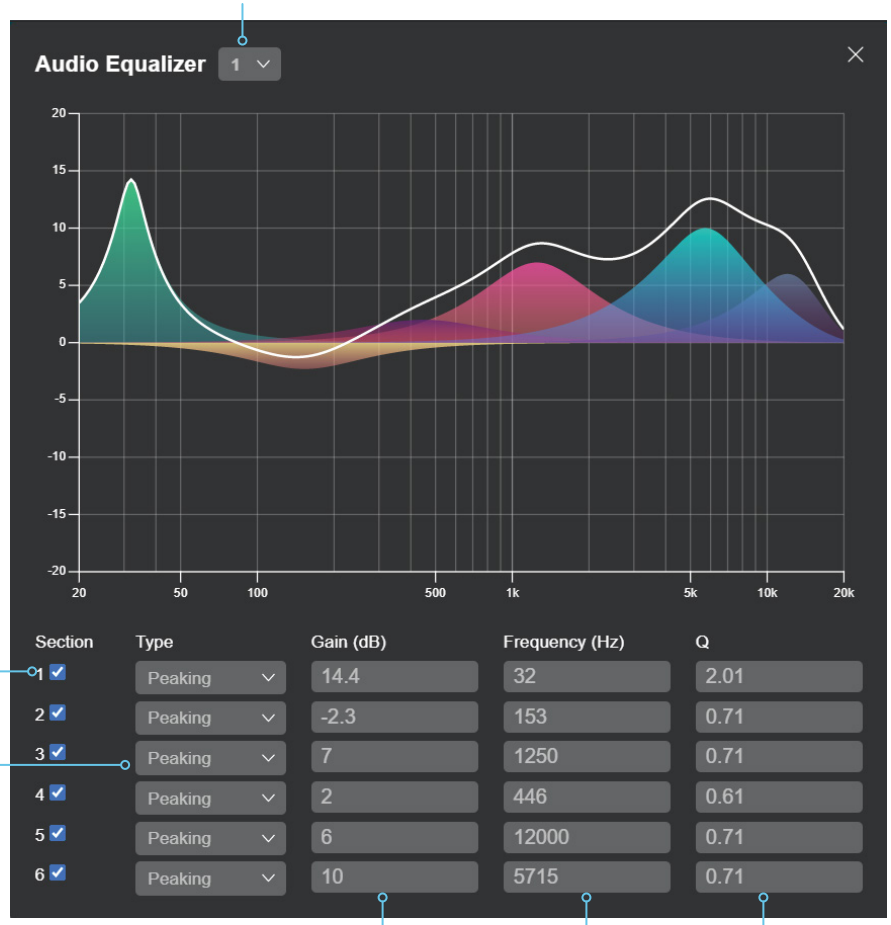
ここまたはここからイコライザーにアクセスできます。



任意のパラメータを変更した場合、Enter キーを押すか別のフィールドを選択するとグラフに表示されます。

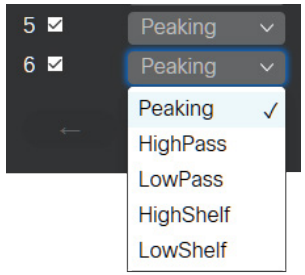
設定は自動的に保存されます。

次の設定に移動します（該当する場合）。



フィルタセクションをアクティブまたは非アクティブにするには、対応するチェックボックスをクリックします。

使用可能なフィルタタイプ



ゲイン設定の値スペースは 0 dB ± 20 dB です。

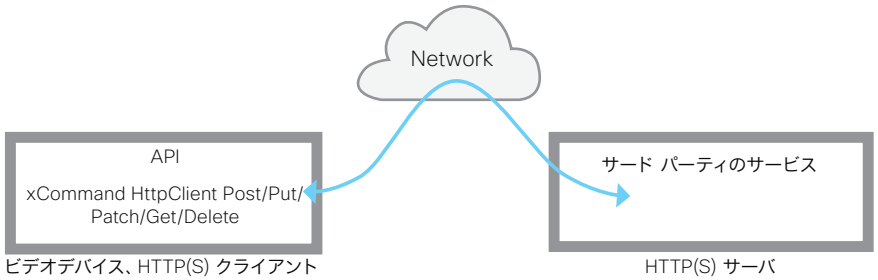
周波数の値スペースは 20 ~ 20 000 Hz です。

Q の値スペースは 0.1~ 10.0 です。

例

HTTPS リクエスト (1/3 ページ)

この機能は任意の HTTP(S) **ポスト**および**プット**リクエストをあるデバイスから HTTP(S) サーバに送信することができます。



API コマンドを使用すると、いつでも HTTP(S) サーバにデータを送信できます。送信するデータを選択して、必要に応じて構造化することができます。この方法で、データをすでに確立されているサービスに適用することが可能です。

セキュリティ対策：

- HTTP(S) **ポスト**/**プット**機能はデフォルトで無効に設定されています。システム管理者は *HttpClient > モード* を *オン*に設定することでこの機能を明示的に有効にする必要があります。
- システム管理者は *HttpClient* t > AllowHTTP を *偽*に設定することで HTTP の使用を防ぐことができます。
- システム管理者は、ビデオデバイスがデータを送信可能な先であるHTTP(S) サーバのリストを指定することができます。詳細については、「`HttpClient Allow Hostname` xCommands」を参照してください。
- 同時に行える `Post` および `Put` 要求の数は制限されています。

許容されている HTTP(S) サーバのリスト

システム管理者はコマンドを使用して最大 10 の許容されている HTTP(S) サーバ (ホスト)のリストを設定し維持できます：

- `xCommand HttpClient` **はホスト名追加表現を許容します：**`<HTTP(S)` **サーバのホスト名または** `IP` **アドレスに一致する正規表現>`**
- `xCommand HttpClient Allow Hostname Clear`
- `xCommand HttpClient Allow Hostname List`
- `xCommand HttpClient Allow Hostname Remove Id: <`**リスト内のエントリーの** `ID>`

リストが空でない場合、HTTP(S) リクエストをリスト内のサーバにだけ送信できます。リストが空の場合、リクエストを任意の HTTP(S) サーバーに送信できます。

許容されているサーバーのリストに対するチェックは、非セキュア (HTTP) およびセキュア (HTTPS) なデータ転送の両方で実行されます。

証明書検証なしの HTTPS の許可

HTTPs 経由で要求を送信する場合、ビデオデバイスはデフォルトで HTTPs サーバの証明書を確認します。HTTPS サーバ証明書が有効でない場合、 エラー メッセージが表示されます。ビデオデバイスはサーバにデータを送信しません。

証明書が検証される HTTPs の使用を推奨します。これが不可能な場合は、システム管理者は *HttpClient > AllowInsecureHTTPS* を *オン*に設定します(`xConfiguration HttpClient AllowInsecureHTTPS: On`)。これにより、HTTPS の使用がサーバ証明書を検証しなくても可能になります。

HTTP リクエスト (2/3 ページ)

HTTP(S) 要求の送信

HTTP(S) クライアントポスト機能が有効になると、以下のコマンドを使用してリクエストを HTTP(S) サーバに送信できます : 以下のテキストで *<メソッド>* は `Post`、`Put`、`Patch`、`Get` **または** `Delete` のいずれかです。

- `xCommand HttpClient` *<メソッド>* `[AllowInsecureHTTPS: <True/False>] [Header:<`**ヘッダーテキスト**`>] [ResponseSizeLimit: <`**最大応答サイズ**`>] [ResultBody: <None/PlainText/Base64>] [Timeout: <`**タイムアウト時間**`>] Url: <`**要求の送信先** `URL>`

ヘッダーフィールドの追加は必須ではありませんが、20 個のフィールドを追加することができます。

`Allowinsecurehttps`パラメータは、システム管理者がサーバの証明書を検証せずに HTTPS の使用を許可した場合にのみ有効です。その場合、パラメータが True に設定されている場合、サーバ証明書を検証せずに、データをサーバに送信できます。パラメータを省略するか、または False に設定すると、証明書の検証が失敗した場合にデータは送信されません。

`ResponseSizeLimit` パラメータは、デバイスがサーバからの応答として受信する最大ペイロードサイズ (バイト) です。応答ペイロードがこの最大サイズより大きい場合、コマンドは、最大ファイルサイズを超えていることを伝えるステータスエラーを返します。ただし、これはサーバ側には影響しません。要求がサーバによって正しく受信され、処理されました。

コマンド結果でサーバから HTTP レスポンスの本文をフォーマットする方法を決定するには、`ResultBody` パラメータを使用します。

次の 3 つのオプションがあります。

- *[なし (None)]*: コマンド結果に HTTP 応答の本体を含めません。
- *Base64*: Base64 は、結果に含める前に本体を base64 エンコードします。
- *プレーンテキスト*: 結果に本文として本文を含める。応答に印字不可能な文字が含まれている場合、コマンドはステータスエラーを返し、印刷できないデータが検出されたというメッセージを返します。

タイムアウトパラメータを使用して、タイムアウト期間 (秒) を設定します。この期間に要求が完了しないと、API はエラーを返します。

コマンドを発行した後に、ペイロード (データ) を直接入力します。改行を含めて、入力したすべてのものがペイロードの一部になります。入力したら、改行 ("\n") と、ピリオドと改行 (".\n") を含む別の行で、終了します。次に、コマンドが実行され、データがサーバに送信されます。

すべてのコマンドと設定については、製品の API ガイドで詳しく説明されています。

HTTP リクエスト (3/3 ページ)

例

例 1: HTTP Post を使用した IoT デバイス制御

次は、Phillips Hue ブリッジに接続されたライトをオンにするマクロ関数を示します。

この例では、メッセージの本文は JSON です。メッセージを受信するサービスの必要な形式に応じて、任意の形式にすることができます。

```
function hue _ command(data) {
  var url = 'http://192.0.2.10/api/'ZXlU4tUtQ23Pjbdyl-kiyCjTs0i5ANDEu1ypJq0-/
lights/1/state';
  var headers = 'Content-Type: application/json';
  var command = '{"on":true}';
  xapi.command('HttpClient Put',
      {'Url': url, 'Header': headers }, command);
}
```

コマンドラインで API を使用して同じ操作を行うことができます。

```
xcommand HttpClient Put Header: "Content-Type: application/json" URL:
"http://192.0.2.10/api/'ZXlU4tUtQ23Pjbdyl-kiyCjTs0i5ANDEu1ypJq0-/lights/1/state"
{"on":true}
```

例 2: HTTP Post を使用したモニタリングツールへのデータの送信

```
xcommand HttpClient Post Header: "Content-Type: application/json" URL:
"https://mymonitoringserver.com/service/devicemonitoring"

{"Message":"A user reported an issue with this system","systemName":"BoardRoom
4th floor","softwareVersion":"ce9.6.0","softwareReleaseDate":"2018-06-
29","videoMonitors":"Dual"}
```

例 3: マクロを使用した HTTP Post の送信

```
function sendMonitoringUpdatePost(message){
    xapi.command('HttpClient Post',
        {'Header': 'Content-Type: application/json',
         'Url':MONITORING _ URL},
          JSON.stringify(Object.assign(
                {'Message': message}, systemInfo)));
}
```

デフォルトボタンの削除 (1/3 ページ)

この機能を使うと、カスタムのコントロール ボタンは UI に表示したまま、デフォルトの機能ボタンを非表示にできるようになります。これにより、UI をカスタマイズできるようになります。

ここでも、xConfigurations を使用するには、デバイスに対するローカル管理者アクセスが必要です。

より具体的には、この機能が行うのは、ビデオのオン/オフを切り替えるボタンなど、これまで非表示にできない特定の機能ボタンを UI 内で非表示/表示できる一連の設定を追加することです。

機能概要

ビデオデバイスの Web インターフェイスに次のコマンドを入力して、設定されている設定を確認します。

```
xconfig //UserInterface/Features
```

使用可能なオプションを表示するには、「?」を追加して同じコマンドを入力します。例：

```
xconfig //UserInterface/Features/ ?
```

たとえば、次が表示されます。

? xConfiguration UserInterface Features Call End: <Auto**, Hidden>*

? xConfiguration UserInterface Features Call JoinWebex: <Auto**, Hidden>*

? xConfiguration UserInterface Features Call Keypad: <Auto**, Hidden>*

? xConfiguration UserInterface Features Call End: <Auto**, Hidden>*

? xConfiguration UserInterface Features Call Start: <Auto**, Hidden>*

? xConfiguration UserInterface Features HideAll: <False**, True>*

? xConfiguration UserInterface Features Share Start: <Auto**, Hidden>*

? xConfiguration UserInterface Features Whiteboard Start: <Auto**, Hidden>*

OK

各行で、デフォルト設定は**太字**で表示されます。「*?」に続くテキストは、実行できるコマンドです。<...> を使用する値に置き換えます。たとえば、*[コール]ボタンを非表示にする場合は、次の操作を実行します。*

```
xConfiguration UserInterface Features Call Start: Hidden
```

[*コール* (Start)] ボタンを非表示にすると、デフォルトの UI 機能が非表示になり、コールを発信したり、ディレクトリルックアップ / お気に入り / 最近の通話を実行したりできます。また、通話中に参加者を追加するために使用される *[追加 (Add)]* ボタンは非表示になります。

Midcallcontrols は*保留*、*転送*、および*再開*されます。

*共有*ボタンを非表示にすると、共有用のデフォルトの UI は非表示になり、ソースをアウトオブコールアウトでプレビューする機能も非表示になります。

制限

この機能は、*コール*、*MidCallControls 制御*、およびボタンの*共有*のセットに対してのみ適用されます。

ミーティング、*エクステンション モビリティ*、*ボイスメール*など、特定の用途の結果を表示する各ボタンを非表示にすることはできません。これらのボタンはすべて表示するか、すべて非表示にする必要があります。

すべてを非表示を選択すると、カスタムボタンのみが表示されます。これを行うには、次のようにします。

```
*? xConfiguration UserInterface Features HideAll: True
```

バックエンドからの設定を選択してプロビジョニングすることによって、このようなすべてまたはなしに発生する問題を回避することができます。

注：この機能はボタンのみを削除することを示しています。関数自体は削除されません。たとえば、*共有*ボタンは削除される場合がありますが、この機能は近接通信によりアクセスできます。

デフォルトボタンの削除 (2/3 ページ)

コールアウトの例

ユーザは、いくつかの特定のルームのみにだけコールするように制限されているシナリオを作成するとします。これは、外部コールを発信しない企業の場合などです。この制限されたルーム数の間ですべてのコールが実行されることを前提としています。

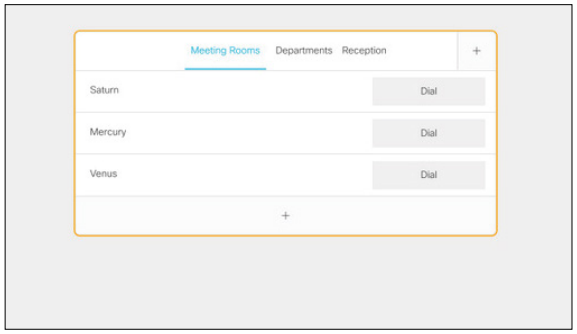
次のような標準の UI を使用して開始します。

次のコマンドを発行するとします：

```
*? xConfiguration UserInterface Features HideAll: True
```

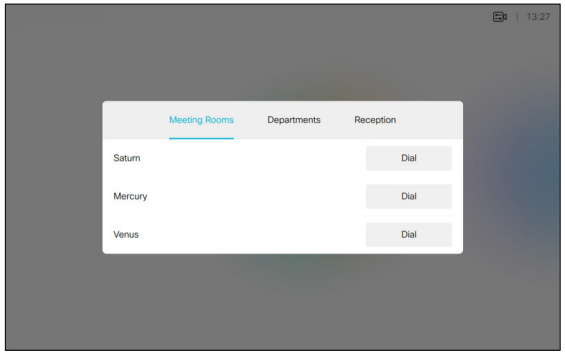
UI は次の様に見えます。

では、**UI 拡張機能エディタ**を開いてこのパネルを作成します。



パネル名は、*Make a call* です。これは、ビデオデバイスにエクスポートするときに、ボタンの下に表示されます。例：

ユーザインターフェイスの *[電話をかける]* ボタンをタップすると、次のパネルが表示されます。



ユーザは、ここで実行できることはあまりありませんが、これは意図的です。誤りまたはランダムコールは不可能です。3 つのルームのいずれかを呼び出すことができるのは、*コール発信*をタップしてから、電話をかけるルームの名称の横にある *[ダイヤル (Dial)]* をタップするだけです。

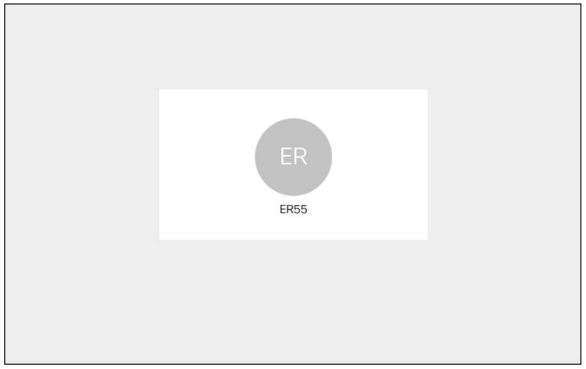
このパネルを実際に作成して使用するには、外部コントロールデバイスを作成するか、**マクロ**を作成する必要があります。マクロの詳細については、このドキュメントの「マクロ」セクションを参照してください。

新しいパネルは通話以外で使用するために使用されたため、通話中にユーザが使用できるパネルまたはボタンを定義する必要があります。そうでない場合は、通話を終了する方法はありません。設定方法が記載されている「通話中の例 」を参照してください。

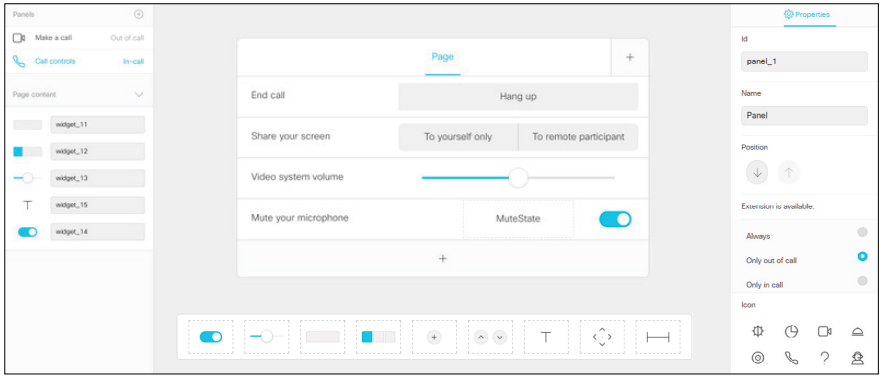
デフォルトボタンの削除 (3/3 ページ)

通話中の例

前のページで説明したように、着信コール動作も定義する必要があります。それ以外の場合は、コール中に次のものを満たします (コールを終了する方法はありません)。

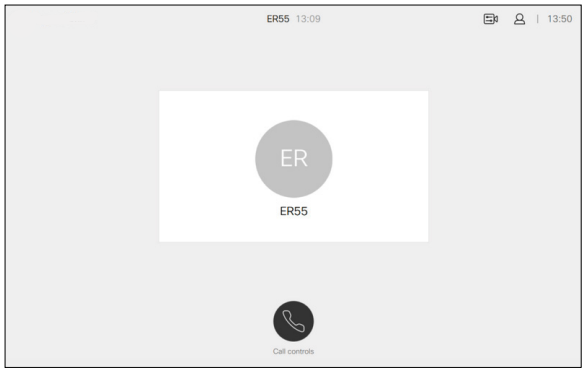


セットアップを作成しましょう。

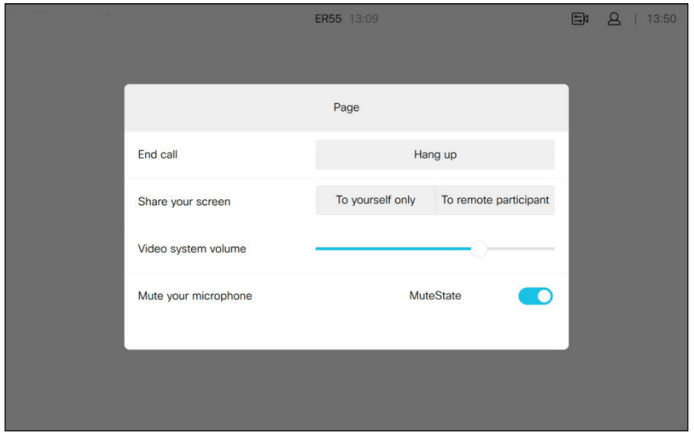


このボタンは、[コール (calls)] にのみ表示されるように設定されます。

これをビデオデバイスにプッシュすると、必要な機能が利用可能になります。



この例では、*[コール制御]* ボタンをタップすると、次のように表示されます。



これらは、作成できる多くのシナリオのほんの一例です。

インタラクティブ メッセージ (1/3 ページ)

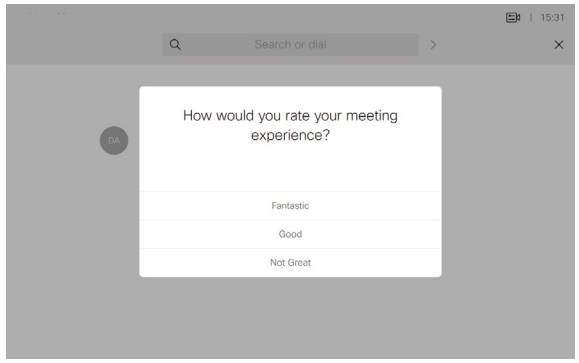
インタラクティブ メッセージ機能を使用すると、ユーザインターフェイスでアラートメッセージおよび/またはインタラクティブ メッセージを作成できます。したがって、ユーザに応じて実行するよう促します。

以前のメッセージでの選択に応じて次のメッセージを表示する、一連のメッセージ群を作成したい場合、そのようなイベント作成を行うマクロの使用を推奨します。または、作成されたイベントに反応する外部のコントロール デバイスを使用することもできます。

ユーザからの入力を送信するには、HttpFeedback の使用に習熟する必要があります。

`HttpFeedback`が原因で、デバイスは API の状態 (ステータス、イベント、設定の更新など) への変更時に、HTTP フィードバック メッセージ (「webhooks」とも呼ばれる) を送信します。HTTP Post フィードバック メッセージは、指定されている ServerURL に送信されます。

例 1： エクスペリエンスを評価する



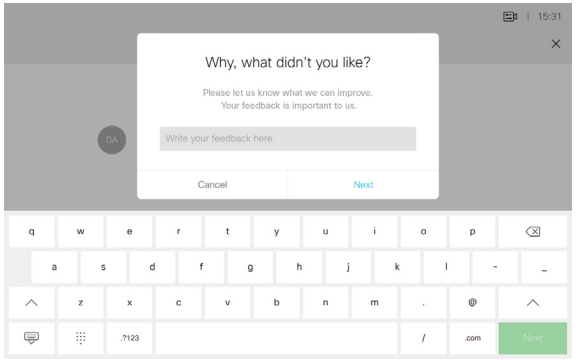

```
xCommand UserInterface Message Prompt Display FeedbackId:"MeetingExperience"
Text:"" Title: "今回の会議エクスペリエンスをどのように評価いただけますか？" Option.1: "すばらしい" Option.2: "良かった" Option.3: "良くなかった"

*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>
```

インタラクティブ メッセージ (2/3 ページ)

例 2: ユーザ入力の要求



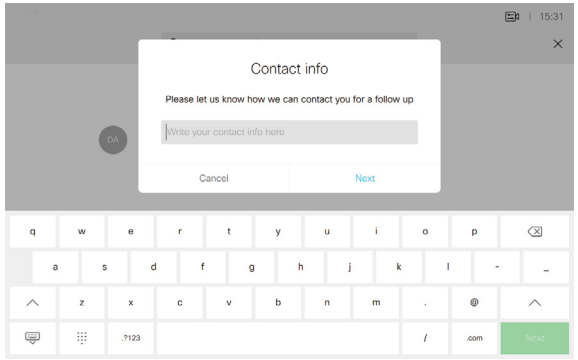

```
xCommand UserInterface Message TextInput Display FeedbackId:
"MeetingFeedback" Placeholder: "ここにご意見をお書きください" SubmitText: "次へ"
Title: "お気に召さなかった点とその理由をお書きください。" Text: "弊社の改善のためのご意見をお
寄せください。お客様のご意見は非常に重要です。"

*e UserInterface Message TextInput Response FeedbackId: "MeetingFeedback"
*e UserInterface Message TextInput Response Text: "解像度が低い"

<XmlDoc resultId="">
<Event>
 <UserInterface item="1">
  <Message item="1">
   <TextInput item="1">
    <Response item="1">
     <FeedbackId item="1">MeetingFeedback</FeedbackId>
     <Text item="1">解像度が低い</Text>
    </Response></TextInput></Message>
 </UserInterface></Event></XmlDoc>
```

ここで示すようにメッセージを作成する場合は、*[次へ (Next)]* ボタンでテキストを指定できます。ただし、*[キャンセル (Cancel)]* ボタンはデフォルトで表示され、テキストは変更できません。

例 3: 連絡を取る



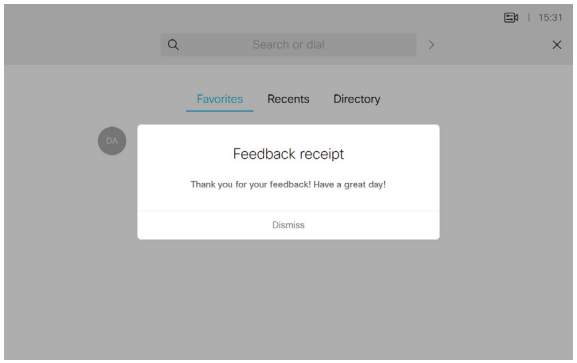

```
xCommand UserInterface Message TextInput Display FeedbackId: "ContactInfo"
Placeholder: "連絡先をご記入ください" SubmitText: "次へ" Title: "連絡先情報" Text: "
フォローアップのためにお客様への連絡手段をお知らせください"

*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>
```

インタラクティブ メッセージ (3/3 ページ)

例 4 : フィードバックの受信



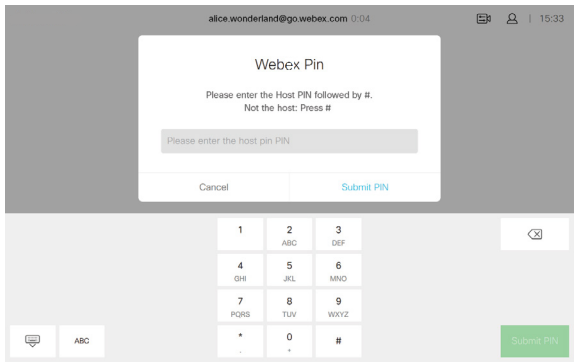

```
xCommand UserInterface Message Alert Display Title: "フィードバック受信" text:
"ご意見を承りました！ありがとうございました。"

*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>
```

メッセージアラートを使用するメッセージを作成する場合、*[破棄 (Dismiss)]* ボタンがデフォルトで表示されます。このボタンのテキストは変更できません。

例 5: Webex 暗証番号の入力




```
xCommand UserInterface Message TextInput Display FeedbackId: "WebexPin"
InputType: Numeric Placeholder: "暗証番号を入力してください" SubmitText: "Submit
PIN" Text: "ホストの PIN をご入力ください。入力が終わりましたら # を押してください。ホストでない場合、# を押してください" 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>
```

サードパーティ製 USB 制御デバイス (1/3 ページ)

この機能は、サードパーティー USB 入力デバイスを使用してビデオデバイスの特定の機能、または挙動を制御することができます。USB キーボードでの Bluetooth® リモート制御 (USB ドングル付き) はこういった入力デバイスの一例です。

注: この機能を使用するには、何らかのアクションを定義して実装する必要があります。このような入力デバイスの使用は、このような操作を行うことを意図したものではありません。アクションの選択にすぐに利用可能なアクションのライブラリがありません。

この機能はタッチ画面を置き換える意味ではありませんが、都合の良い場所では機能の一部を補完する場合があります。

この機能の目的は次のとおりです。

この機能用のアプリケーションの例は、通常、次のようになります。

- 教室内および講義中は、小規模なリモートコントロールを使用して、ビデオシステムをスタンバイ モードから起動し、表示する入力ソースを選択できます。
- [Touch 10] が不便である、または使用を許可されていない場合において、カメラのパン、傾き、およびズームを制御します (たとえば、遠隔医療と組み合わされた病院の手術室の場合)。

実際の例は、「サードパーティ USB 入力デバイスの使用例」参照してください。

入力デバイスの要件

入力デバイスは、USB キーボードとしてアドバタイズされていなければなりません。キーボードという用語は、必ずしもここでは理解する必要はありません。USB ドングルを使用した Bluetooth リモートコントロールがこの操作を実行します。

機能概要

USB 入力デバイスのボタンを押すと、API でイベントが生成されます。マクロまたはサードパーティーの制御デバイスは、こういったイベントをリッスンして応答することが可能です。これは、Touch 10 ユーザインターフェイスのボタンと似ています。Webhook を使って、SSH セッションで直接イベントをリッスンすることも可能です。

使用者がイベントへの応答として実行するアクションを定義し、実装する必要があります。次に例を示します。

- *[音量アップ (Volume Up)]* キーが押されたら、ビデオデバイスの音量を上げます。
- *[スリープ (Sleep)]* キーが押されたら、ビデオデバイスは [スタンバイ (Standby Mode)] モードになります。

注: USB 入力デバイスのサポートはデフォルトで無効になっています。これは、明示的に有効にする必要があることを意味します。*周辺機器 > InputDevice > Modeをオン*に設定します。

ボタンを押してから離すと、**押された**イベントおよび**リリースされた**イベントが作成されます:

```
*e UserInterface InputDevice Key アクション Key: <キーの名前>
*e UserInterface InputDevice Key アクション コード: <キーの ID>
*e UserInterface InputDevice Key Action Type: Pressed
** end
*e UserInterface InputDevice Key アクション Key: <キーの名前>
*e UserInterface InputDevice Key アクション コード: <キーの ID>
*e UserInterface InputDevice Key Action Type: Released
** end
```

イベントをリッスンするには、InputDevice イベントからのフィードバックを登録する必要があります。

```
xFeedback Register /event/UserInterface/InputDevice
** end
```

ビデオデバイスによって入力デバイスが検出されると、*UserInterface > 周辺機器 > connecteddevice* ステータスに表示されます。入力デバイスは複数のデバイスとして報告される場合があります。

サードパーティ製 USB コントロールデバイス (2/3 ページ)

サードパーティ USB 入力デバイスの使用例

この例では、サードパーティの USB 入力デバイス (リモート制御など) のキーを使用する方法を示しています。スタンバイ機能の操作、音量の増減、カメラの操作ができます。

作成されたマクロは、関連するイベントをリスンし、ルームまたはデスクデバイスの API を使用して、関連付けられたアクションを実行します。

注: 下のコマンド例では、ユーザ入力のテキストは標準フォントで表示されています。斜体のテキストは、ルーム デバイスから受信する応答です。

1. SSH 上のルームまたはデスクデバイスにログインします。ローカルの管理者ユーザが必要です。
2. サードパーティ USB リモート制御の使用を許可するようにデバイスを設定します。

```
xConfiguration Peripherals InputDevice Mode: On

** end

[OK]
```

注: このコマンドを使用すると、構成が *On* または *Off* かを確認できます。

```
xConfiguration Peripherals InputDevice Mode

* c Xxconfiguration Peripherals InputDevice Mode: On

** end

[OK]
```

3. フィードバックに登録して、リモートコントロールボタンが押されて離されたときに通知されるようにします。

```
xFeedback Register /event/UserInterface/InputDevice

** end

[OK]
```

注: このコマンドを使用して、デバイスがどのフィードバックに登録されているかを確認できます。

```
xFeedback list

/event/UserInterface/inputDevice

** end

[OK]
```

4. リモートコントロールのボタンを押して放し、フィードバック登録が機能することを確認します。

   これにより、**押した状態**および**リリース済みの 2 つの異なるイベントが生成されます。**ボタンを押したままにすると、ボタンが離されるまで、**押された**イベントが表示されます。その後、**リリース済み**のイベントが生成されます。

   次のイベントは、`Enter` キーを押して離したときに発行されます。

```
*e UserInterface InputDevice Key アクション Key: KEY _ ENTER

*e UserInterface InputDevice Key アクション コード: 28

*e UserInterface InputDevice Key Action Type: Pressed

** end
```

5. 関連する *InputDevice* イベントをリッスンし、デバイスの API を使用して関連するアクションを実行するマクロを作成します (これは次のページに表示されます) 。

   a. [スタンバイ (standby)]、[音量 (up)]、[音量ダウン] ボタンを有効にします。

   マクロが `KEY _ VOLUMEUP`、`KEY _ VOLUMEDOWN`、または`KEY _ SLEEP`を含むイベントを表示する場合は、関連コマンドを実行します。

   b. 矢印キー用のカメラ制御機能を作成します。ボタンが押されている間は、カメラの移動を維持する必要があります。ボタンを離すと、カメラの動きが停止します。

   マクロは、`KEY _ LEFT`、`KEY _ RIGHT`、`KEY _ UP`、または`KEY _ DOWN`を含むイベントを検出すると、関連コマンドを実行します。

サードパーティ製 USB コントロールデバイス (3/3 ページ)

カメラ制御機能に関連する部品は、以下の青色で識別されます。

```
const xapi = require('xapi');
function com(command, args='') {
  xapi.command(command, args);
  log(command + ' ' + JSON.stringify(args));
}
function log(event) {
  console.log(event);
}
function notify(message) {
  xapi.command('UserInterface Message TextLine
Display', {
    Text: message,
    duration: 3
  });
}
function cameraControl(motor, direction,
cameraId='1') {
  com('Camera Ramp', { 'CameraId': cameraId,
                       [motor]: direction
  });
}
function init() {
  let standbyState;
  xapi.status.get('Standby').then((state) =>
{standbyState = state.State === 'Off' ? false :
true; });
  xapi.status.on('Standby', state => {
      standbyState = state.State === 'Off' ? false
: true;
  });
    xapi.event.on('UserInterface InputDevice Key
Action', press => {
    if (press.Type == "Pressed") {
      switch (press.Key) {
        case "KEY_LEFT":
          cameraControl('Pan', 'Left');
          break;
        case "KEY_RIGHT":
          cameraControl('Pan', 'Right');
          break;
        case "KEY_UP":
          cameraControl('Tilt', 'Up');
          break;
        case "KEY_DOWN":
          cameraControl('Tilt', 'Down');
          break;
        デフォルト:
          break;
      }
    } else if (press.Type == "Released") {
      switch (press.Key) {
        case "KEY_LEFT":
          cameraControl('Pan', 'Stop');
          break;
        case "KEY_RIGHT":
          cameraControl('Pan', 'Stop');
          break;
        case "KEY_UP":
          cameraControl('Tilt', 'Stop');
          break;
        case "KEY_DOWN":
          cameraControl('Tilt', 'Stop');
          break;
        case 'KEY_VOLUMEUP':
          com('Audio Volume Increase');
          break;
        case 'KEY_VOLUMEDOWN':
          com('Audio Volume Decrease');
          break;
        case 'KEY_SLEEP':
          com(standbyState ? 'Standby
Deactivate' : 'Standby Activate');
          break;
        デフォルト:
          break;
      }
    }
  });
}
init();=
```

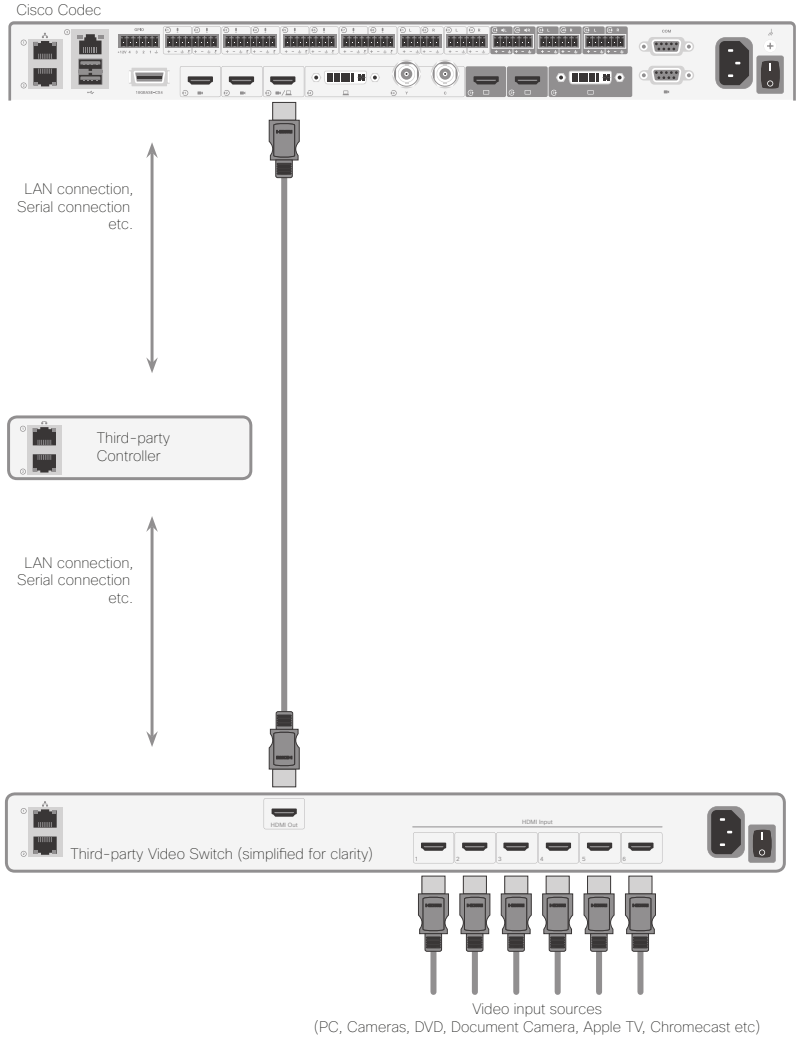
ビデオ スイッチの使用 (1/4 ページ)

サードパーティ製ビデオ スイッチを使って使用可能なビデオ ソース数を拡張する

UI 拡張機能エディタでは、サードパーティの外部ビデオ スイッチからのビデオ ソースを通常の *[共有トレイ (Share Tray)]* ビューに表示するよう設定できます。

ソースは、ビデオ会議デバイスに直接接続されている他のビデオデバイスと同じように表示されて動作します。ユーザには、完全に透過的である、つまりビデオ スイッチが使用されていないように見えます。

設定を使用したセットアップの「ビデオ スイッチの例」を参照してください。



ビデオスイッチの使用 (2/4 ページ)

API コマンドとイベント

ビデオ スイッチ機能には、サードパーティ製の制御システムが必要です。制御システムは API を使用し、いくつかの API イベントとコマンドを使ってビデオ スイッチとお使いのシステムのユーザ インターフェイスの間でソース状態を同期させます。

ユーザがビデオソースを選択するときにこれを動作させるには、対応するイベントを発行するためにビデオデバイスを設定する必要があります。これにより、コントローラは適切なコマンドをビデオスイッチとビデオデバイスに送信します。

このイベントは、コントローラが接続時にビデオデバイスに登録し、ビデオデバイスに次を要求した場合にのみ発行されます。

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

発行されるイベントは次のとおりです。

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

ここでの "XXXX は、状態を選択または設定するときにこのソースを識別するための固有の文字列 ID です。

システムを制御するために次の 6 つのコマンドを使用できます。

Add：ビデオ ソース識別子を追加します。これにはコネクタの ID、画面に表示される名前、状態を選択または設定するときにソースを識別するための固有文字列 ID、ソースごとに画面に表示させるアイコンの種類が含まれます。

List：現在の外部ソースのリストを返します。

Remove：リストからソースを削除します。

RemoveAll：リストからすべてのソースを削除します。

Select：特定のソースを選択します。

State Set：ソースの状態を変更します。

詳細については、UserInterface Presentation ExternalSource API コマンドを参照してください。

ビデオスイッチの使用 (3/4 ページ)

UserInterface Presentation ExternalSource API コマンド

UserInterface Presentation ExternalSource Add コマンド

このコマンドは入力ソースを確立して定義します。

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

引数:

ConnectorId: 外部スイッチが接続されているビデオ デバイス コネクタの ID。

名前 : 画面に表示されている名前

SourceIdentifier: 状態を選択または設定するときにこのソースを識別するための固有の文字列 ID

Type: 画面に表示されるアイコンを決定します。次から選択してください : <pc/camera/desktop/document_camera/mediaplayer/other/whiteboard>

例:

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

UserInterface Presentation ExternalSource List コマンド

このコマンドは、現在の外部ソースのリストを返します。

```
xcommand UserInterface Presentation ExternalSource List
```

UserInterface Presentation ExternalSource Remove コマンド

このコマンドは、リストからソースを削除します。

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

引数:

*SourceIdentifier*は、状態を選択または設定するときにこのソースを識別するための固有の文字列 ID です。

UserInterface Presentation ExternalSource RemoveAll コマンド

このコマンドは、リストからすべてのソースを削除します。

```
xCommand UserInterface Presentation ExternalSource RemoveAll
```

UserInterface Presentation ExternalSource Select コマンド

選択したソースが準備完了状態で、有効な ConnectorId を持っている場合、選択されたソースを表示し始めます。また、アイテムを共有トレイに「表示中 (Presenting) 」として表示します。

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

引数:

*SourceIdentifier*は、状態を選択または設定するときにこのソースを識別するための固有の文字列 ID です。

UserInterface Presentation ExternalSource State Set コマンド

SourceIdentifier を持つソースの状態を変更するために使われます。

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

引数:

SourceIdentifier は、状態を選択または設定するときにこのソースを識別するための固有の文字列 ID です。

State : <Error/Hidden/NotReady/Ready> 表示可能な状態は Ready (準備完了) のみです。リストには Hidden (非表示) が含まれていますが、共有トレイには表示されません。

ErrorReason : オプション。状態が `Error` に設定されている場合に、共有トレイに表示されます。

ビデオスイッチの使用 (4/4 ページ)

ビデオ スイッチの例

コントローラが次のものを送信します：

```
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
```

デフォルト状態は `NotReady` です (図 1)

インテグレータは次のステップとして、これらを Ready に設定できます (図 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
```

ビデオ スイッチでいずれかのソースが選択された場合、コントローラはそれに応じてコマンドを送信する必要があります：

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

選択されたコネクタでスイッチが接続されると、表示が始まります (図 3) 。

ユーザが共有トレイで別のソース項目をクリックして別のソースを選択すると、ビデオデバイスは次のイベントを送信します。

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

コントローラはこのイベントをリッスンして、選択されたソースを表示する必要があります。

注：以下の設定が**マニュアル**になっている場合、プレゼンテーションは開始されません。

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

図 1：デフォルト状態は NotReady。

図 2：入力ソースが Ready に設定されたところ。

図 3：選択されたコネクタでスイッチが接続されると、表示が始まる。

ルーム制御の戦略例

照明の制御

スライダとトグル ボタンを組み合わせて使用し、照明を制御することができます。トグル ボタンで照明のオン/オフを切り替えて、スライダを調光器として使用します。

次の方法を検討します。

- ユーザが照明をオフにしたら、スライダを最小レベルに設定します。
- ユーザがスライダを最小レベルに移動したら、トグル ボタンをオフに設定します。
- 照明がオフになったときのスライダの値を保存しておき、再びオンになったときに元の値に戻すことができるようにします。
- 明るさが 40 % のときに照明をオフにして、もう一度オンにすると、ユーザは照明が (最大の明るさではなく) 40 % の明るさになると予想します。
- ユーザがグループ ボタン (照明のプリセット) でオプションの 1 つを選択したら、それに応じてスライダとトグル ボタンを設定します。
- たとえば、スライダまたはトグル ボタンを操作して照明がプリセットから変更された場合は、グループ ボタンのすべてのオプションを選択解除します。

温度の制御

スピン ボックスとフォントが大きいテキスト ボックス (値) を組み合わせて使用し、温度を制御することができます。スピン ボックスを使用して希望の温度を設定し、フォントが大きいテキスト ボックスに現在の温度を表示します。

最良のユーザ エクスペリエンスを実現するには、以下の点を念頭に置いてください。

- 室温が変化したら、フォントが大きいテキスト ボックスの表示内容を更新します。上矢印や下矢印がタップされたら、スピン ボックスのテキスト フィールドを更新します。
- スピン ボックスのテキスト フィールドとフォントが大きいテキスト ボックスを更新する方法の詳細については、「ウィジェット (Widgets)」の章を参照してください。

ブラインドの制御

スピン ボックスを使用するか、またはウィジェット ライブラリの [アイコン (Icons)] タブから上矢印と下矢印を使用することができます。

次の方法を検討します。

- 方向矢印を短く押すと、ブラインドが傾きます。すでに最大限傾いている場合は、ブラインドが段階的に上下に動きます。
- 方向矢印を長押しすると、ブラインドがその方向に動き始めます。完全に上がるか下がるまで停止しません。
- 長押し後の動きを止めるには、いずれかのボタンを短く押します。別個の停止ボタンは必要ありません。

オンラインリソース

マクロと拡張機能の例

▶ https://roomos.cisco.com

YouTube

▶ https://www.youtube.com/playlist?list=PL_YnWo4XhzTe8CyOppyhheB8UN40mDYiX

xAPI サンプル

▶ https://github.com/CiscoDevNet/roomdevices-macros-samples

awesome-xapi: 厳選されたコミュニティリソース

▶ https://github.com/CiscoDevNet/awesome-xapi

知的財産

このマニュアルに記載されている仕様および製品に関する情報は、予告なしに変更されることがあります。このマニュアルに記載されている表現、情報、および推奨事項は、すべて正確であると考えていますが、明示的であれ黙示的であれ、一切の保証の責任を負わないものとします。このマニュアルに記載されている製品の使用は、すべてユーザー側の責任となります。

対象製品のソフトウェア ライセンスと限定保証は、製品に添付された『Information Packet』に記載されています。添付されていない場合には、代理店にご連絡ください。

Cisco が採用している TCP ヘッダー圧縮機能は、UNIX オペレーティング システムの UCB (University of California, Berkeley) のパブリック ドメイン バージョンとして、UCB が開発したプログラムを採用したものです。All rights reserved. Copyright © 1981, Regents of the University of California.

ここに記載されている他のいかなる保証にもよらず、各社のすべてのマニュアルおよびソフトウェアは、障害も含めて「現状のまま」として提供されます。シスコおよび上記代理店は、商品性、特定目的適合、および非侵害の保証、もしくは取り引き、使用、または商慣行から発生する保証を含み、これらに限定することなく、明示または暗黙のすべての保証を放棄します。

いかなる場合においても、シスコおよびその供給者は、このマニュアルの使用または使用できないことによって発生する利益の損失やデータの損傷をはじめとする、間接的、派生的、偶発的、あるいは特殊な損害について、あらゆる可能性がシスコまたはその供給者に知らされていても、それらに対する責任を一切負わないものとします。

このマニュアルで使用している IP アドレスおよび電話番号は、実際のアドレスおよび電話番号を示すものではありません。マニュアルの中の例、コマンド出力、ネットワーク トポロジ図、およびその他の図は、説明のみを目的として使用されています。説明の中に実際の IP アドレスおよび電話番号が使用されていたとしても、それは意図的なものではなく、偶然の一致によるものです。

印刷版と複製ソフトは公式版とみなされません。最新版はオンライン版を参照してください。

シスコは世界各国 200 箇所にオフィスを開設しています。各オフィスの住所、電話番号、FAX 番号は当社の Web サイト (www.cisco.com/go/offices) をご覧ください。

Cisco および Cisco ロゴは、Cisco Systems, Inc. またはその関連会社の米国およびその他の国における登録商標または商標です。シスコの商標の一覧については、https://www.cisco.com/c/ja_jp/about/legal/trademarks.html をご覧ください。Third-party trademarks mentioned are the property of their respective owners. 「パートナー」という用語の使用は Cisco と他社との間のパートナーシップ関係を意味するものではありません。(1110R) 。

シスコのお問い合わせ先

Cisco のウェブサイトでは、Cisco の世界各地のお問い合わせ先を確認できます。

移動先 : ▸ https://www.cisco.com/go/offices

本社

Cisco Systems, Inc.

170 West Tasman Dr.

San Jose, CA 95134 USA