Video Conferencing Using the Session Initiation Protocol Client

Appendix Overview

The Session Initiation Protocol (SIP) client enables a customer at the kiosk to make a SIP audio and video call with a remote assistant.

Topics in this appendix include:

- “SIP Recommendations”
- “SipPhone Widget”
  - “Sample Test Code”
- “IEC Preparation”
- “SIP Client”
- “Cisco IP Phone Set Up on the CUCM”
  - “Finding The IP Phone’s MAC Address”
- “Cisco IEC Set Up on the CUCM”
- “Configuring Call Manager Information”
  - “Using a Policy on the IEM”
  - “Using the SipPhone Widget”
- “SIP DTMF”
  - “Sample usage of sendDtmf() API”

SIP Recommendations

The following are recommendations when using SIP:

- SIP video quality is dependent on the available network link. At least 1Mbps of available bandwidth between the end-points is recommended for HD-quality video call.
- Since HD quality is affected greatly by poor network design, it is recommended that the network link is not congested.
- When using the SIP widget with another video application such as the video player, ensure that all videos have stopped when SIP receives an incoming signal.
- Use an USB external microphone and USB speakers to get the best result for echo cancellation.
- Use a recommended camera for HD quality video such as the Cisco PrecisionHD camera or Logitech C920 camera.

**SipPhone Widget**

Cobra provides several proprietary widgets to simplify developer's life. Those widgets can be configured and controlled from JavaScript. The sipphone widget allows you to make SIP phone calls to another SIP endpoint. This plugin acts like a True SIP endpoint and supports both audio and video calls. Both SD (g711) and HD (g7221) audio codecs are supported. For video, it supports H.263 and H.264 codecs.

The sipphone interface declaration is:

```javascript
interface SipPhone
{
    attribute int height;
    attribute int width;
    attribute string backgroundColor;
    attribute string idleImage;
    attribute bool videoEnabled; // Is true by default.
    attribute string status;

    slots:
    int start (in string username,in string password,in string domain,in string transport);
    void call(in string sipUri);
    void hangup();
    void sendDtmf(in string dtmfkey);
    bool setIdleImage(in string imgurl, in bool stretchFlag);
    bool changeIdleImage(in string imgurl, in bool stretchFlag);
    string cameraDevice() const;
    int setCameraDevice(in string deviceId);
    int capture() const;
    string getImage() const; // Returns the Jpeg image if captured
    void answer();
    void reject();
    void setAutoAnswer(in bool autoAnswerFlag);

    signals:
    void ready();
    void registered();
    void placingCall();
    void incomingCall();
    void established();
    void ring();
    void disconnected();
    void video();
    void noVideo();
    void hold();
    void resume();
    void captured();
    void error(in int code, in string explanation);
};
```
## Table F-7 sipphone Variables

<table>
<thead>
<tr>
<th>Method Call</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>start(string username, string password, string domain, string transport)</code></td>
<td>This method call is to be used to set the SIP credentials that are needed to get registered with the SIP Registrar (or Call Manager). The needed credentials are Username, Password, Domain (IP Address or Domain Name of the SIP Registrar) and the transport to be used (UDP or TCP).</td>
</tr>
<tr>
<td><code>call(string sipUri)</code></td>
<td>This method should be used only after the start(...) method is called. This method initiates the call to the sipUri (called party).</td>
</tr>
<tr>
<td><code>hangup()</code></td>
<td>This method, when called, disconnects the existing call.</td>
</tr>
<tr>
<td><code>sendDtmf(string dtmfkey)</code></td>
<td>This method sends DTMF tones to the SIP proxy. Valid DTMF keys are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, *, and #.</td>
</tr>
<tr>
<td><code>setIdleImage(string imgUrl, bool stretchFlag)</code></td>
<td>This method can be used to display an image, like logo or some graphic when the SIP widget is registered and not in a call. This method provides a mechanism for the widget to display an image when it is not in a call. The parameters are imgUrl, the URL for the image to be displayed, and stretchFlag, which indicates whether to auto resize or not the image to the given frame.</td>
</tr>
<tr>
<td><code>changeIdleImage(string imgUrl, string sipUri)</code></td>
<td>This method is similar in functionality to setIdleImage. You could use this method to change the appearance of the widget like coding it in Javascript to change the idleimage to create the sense of screen saver for the widget.</td>
</tr>
<tr>
<td><code>cameraDevice()</code></td>
<td>This method returns the currently configured webcam that is being used by the SipPhone widget. The value returned would be in the UNIX format similar to “/dev/video0”.</td>
</tr>
<tr>
<td><code>setCameraDevice()</code></td>
<td>Use this method to let the SipPhone widget know which webcam to use to place the call. You need to call this API with UNIX format identifier for camera, such as “/dev/video0” or “/dev/video1”.</td>
</tr>
<tr>
<td><code>capture()</code></td>
<td>Use this method to initiate taking a still image when the video call is in progress. This is useful if you would like to take a snapshot of the participant and save it for future reference.</td>
</tr>
</tbody>
</table>

**Note**
Call this API before the start() method in the Javascript.

**Caution**
Call this routine only when there is an active video call.
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getImage()</td>
<td>Call this method after you have received a captured() signal. When called, this routine returns base64 content of the JPEG image captured.</td>
</tr>
<tr>
<td>answer()</td>
<td>Accepts incoming call.</td>
</tr>
<tr>
<td>reject()</td>
<td>Rejects incoming call.</td>
</tr>
<tr>
<td>setAutoAnswer(in bool autoAnswerFlag)</td>
<td>Enables auto answer mode if the autoAnswerFlag is “true”.</td>
</tr>
<tr>
<td>ready()</td>
<td>This signal is indicative that values given for initializing the SIP phone are accepted.</td>
</tr>
<tr>
<td>registered()</td>
<td>This signal means that the SIP phone is now registered with the SIP Registrar (or Call Manager) and you can make and receive calls from the widget.</td>
</tr>
<tr>
<td>placingCall()</td>
<td>This signal means the widget is trying to place the call to the called party of interest.</td>
</tr>
<tr>
<td>incomingCall()</td>
<td>This signal means the widget is receiving an incoming call request from another SIP peer.</td>
</tr>
<tr>
<td>established()</td>
<td>This signal is indicative that the call is in progress.</td>
</tr>
<tr>
<td>ring()</td>
<td>This signal means that the called party has been notified about the incoming call.</td>
</tr>
<tr>
<td>disconnected()</td>
<td>This signal means that the call has been terminated.</td>
</tr>
<tr>
<td>video()</td>
<td>This signal means that the call was negotiated as a video call and the remote site video is available to display.</td>
</tr>
<tr>
<td>novideo()</td>
<td>This signal means that the call that was negotiated does not have video being sent by the remote end. The application can use the novideo signal to improve the user experience such as displaying a “Please wait” message.</td>
</tr>
<tr>
<td>hold()</td>
<td>This signal means that the remote party has put the call on hold. An image can be displayed on the screen when a SIP call is placed on hold. The image that is included in this signal will be shown on the screen.</td>
</tr>
<tr>
<td>resume()</td>
<td>This signal means that the remote party has resumed the call. Upon receiving this signal, the application will revert to the original screen and the on-hold image will be hidden.</td>
</tr>
</tbody>
</table>
Sample Test Code

<!-- IEC-4.155.393 -->
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>SIP phone</title>
<style type="text/css">
html, body {padding:0; margin:0; width:100%; height:100%}
body (background: #1e2024; color: #ffffff; font: normal 12px Arial, Helvetica, sans-serif;)
body (background: -webkit-gradient(linear, left top, left bottom, color-stop(0%,rgb(180,180,177,1)), color-stop(32%,rgb(214,213,212,1)), color-stop(100%,rgb(255,255,255,1))); background: -webkit-linear-gradient(top, rgba(180,180,177,1) 0%,rgba(214,213,212,1) 32%,rgba(255,255,255,1) 100%);)
ul, ol, li (padding:0; margin:0;)
.vbox, .hbox {
display: -webkit-box;
display: box;
-webkit-box-pack: justify;
box-pack: justify;
text-align:justify;
text-align:center;
}
.vbox {
-webkit-box-orient: vertical;
box-orient: vertical;
}
.hbox {
-webkit-box-orient: horizontal;
box-orient: horizontal;
}
.fullwindow {
height:100%; width:100%;
display: -webkit-box;
display: box;

<table>
<thead>
<tr>
<th>captured()</th>
<th>This signal is fired when an user action of taking a still snapshot of an video call is successfully finished. Once this signal is fired, the application can call getImage() to get the captured image (as a JPEG image).</th>
</tr>
</thead>
</table>
| error(int code, in string explanation) | This signal is indicative of any errors whilst the widget operation. The signal has a code and an explanation about the error that was encountered.  
**Error Codes:**  
404: No answer  
401: Registration failed  
485: User busy  
494: User not found  
486: Call manager not able to route call  
503: Service unavailable |

Sample Test Code

```html
<!-- IEC-4.155.393 -->
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>SIP phone</title>
<style type="text/css">
html, body {padding:0; margin:0; width:100%; height:100%}
body (background: #1e2024; color: #ffffff; font: normal 12px Arial, Helvetica, sans-serif;)
body (background: -webkit-gradient(linear, left top, left bottom, color-stop(0%,rgb(180,180,177,1)), color-stop(32%,rgb(214,213,212,1)), color-stop(100%,rgb(255,255,255,1))); background: -webkit-linear-gradient(top, rgba(180,180,177,1) 0%,rgba(214,213,212,1) 32%,rgba(255,255,255,1) 100%);)
ul, ol, li (padding:0; margin:0;)
.vbox, .hbox {
display: -webkit-box;
display: box;
-webkit-box-pack: justify;
box-pack: justify;
text-align:justify;
text-align:center;
}
.vbox {
-webkit-box-orient: vertical;
box-orient: vertical;
}
.hbox {
-webkit-box-orient: horizontal;
box-orient: horizontal;
}
.fullwindow {
height:100%; width:100%;
display: -webkit-box;
display: box;
```
/*============ SIP ================*/
.LED_red, .LED_green, .LED_white, .LED_off {width: 44px; height: 24px;
background: transparent url('img/LED_off.png') center center no-repeat;}
.LED_red {background-image: url('img/LED_red.png');}
.LED_green {background-image: url('img/LED_green.png');}
.LED_white {background-image: url('img/LED_white.png');}
.topPanel {background: #0e1014 url('img/top_panel_bg.png') top left repeat-x;
color: #4c5058; text-shadow: 0 -1px 1px #000; font: normal 14px Arial, Helvetica, sans-serif; height: 41px;}
.status {color: #848d9d; text-shadow: 0 -1px 1px #000; font: normal 14px Arial, Helvetica, sans-serif; padding: 0 20px}
.bottomPanel {background: #0e1014 url('img/bottom_panel_bg.png') top left repeat-x;
color: #4c5058; text-shadow: 0 -1px 1px #000; font: normal 14px Arial, Helvetica, sans-serif; height: 80px;}
.calltime {color: #b1b6c3; font: normal 14px Arial, Helvetica, sans-serif; height: 41px; text-shadow: 0 -1px 1px #000;}
.buttons {text-align: center; display: inline-block; padding: 0 30px 0 30px;}
.greenButton, .redButton, .callButton, .endCallButton, .hangupButton, .acceptButton, .rejectButton {width: 170px; height: 56px; background: transparent;
-webkit-border-image: url('img/greenButton_disabled.png') 1 10 1 10 stretch;
border-width: 1px 10px 1px 10px; color: rgba(255, 255, 255, .9); font: normal 22px Arial, Helvetica, sans-serif; padding-top: 6px; text-shadow: 0 -1px 1px rgba(0, 0, 0, .6)}
.greenButton, .callButton, .acceptButton { -webkit-border-image: url('img/greenButton_idle.png') 1 10 1 10 stretch stretch;}
.greenButton:active, .callButton:active, .acceptButton:active { -webkit-border-image: url('img/greenButton_pressed.png') 1 10 1 10 stretch stretch;}
.greenButton:disabled, .callButton:disabled, .acceptButton:disabled { -webkit-border-image: url('img/greenButton_disabled.png') 1 10 1 10 stretch stretch; color: #51565d}
.redButton, .endCallButton, .hangupButton, .rejectButton { -webkit-border-image: url('img/redButton_idle.png') 1 10 1 10 stretch stretch;}
.redButton:hover, .endCallButton:hover, .hangupButton:hover, .rejectButton:hover { -webkit-border-image: url('img/redButton_hover.png') 1 10 1 10 stretch stretch;}
.redButton:active, .endCallButton:active, .hangupButton:active, .rejectButton:active { -webkit-border-image: url('img/redButton_pressed.png') 1 10 1 10 stretch stretch;}
.redButton:disabled, .endCallButton:disabled, .hangupButton:disabled, .rejectButton:disabled { -webkit-border-image: url('img/redButton_disabled.png') 1 10 1 10 stretch stretch; color: #51565d}
.view {background: #000; border: solid 1px #3b3d40; text-align: center; width: 640px; height: 380px;}
.timerOn, .timerOff {color: #b1b6c3; font: normal 36px Arial, Helvetica, sans-serif; height: 41px; text-shadow: 0 -1px 1px #000;}
.timerOff {color: #292e33}
.dialpad_disabled {width: 43px; height: 43px; background: transparent url('img/dialpad_disabled.png') center center no-repeat; border: none}
</style>
<script>
var sipphone, sipButton, sipRejectButton, sipTimer, sipStatus, sipRegistered;
var calllnProgress;
var sip_target = "133";
var sip_username = "vep1";
var sip_password = "user132resu";
var sip_domain = "192.168.0.108";
var sip_transport = "udp";
var useApplicationData = true;
function initSIP(){
    writeLog('Starting SIP widget:');
    callInProgress = false;
    sipphone = document.getElementById("sipphone");
    sipbutton = document.getElementById("CallButton");
    sipRejectButton = document.getElementById("RejectButton");
    sipTimer = document.getElementById("callTimer");
    sipRegistered = document.getElementById("registeredLED");
    sipStatus = document.getElementById("SIPstatus");
    var default_target = sip_target;
    var default_username = sip_username;
    var default_password = sip_password;
    var default_domain = sip_domain;
    var default_transport = sip_transport;
    if(useApplicationData){
        writeLog('Using application data.');
        // These are the Credentials for the SIP endpoint
        // It is recommended that you use the
        // Application Data at the IBM profile to set these
        // Values and get them via the global.applicationData.value() API.
        sip_target = global.applicationData.value("sip.target", default_target);
        sip_username = global.applicationData.value("sip.username", default_username);
        sip_password = global.applicationData.value("sip.password", default_password);
        sip_domain = global.applicationData.value("sip.domain", default_domain);
        sip_transport = global.applicationData.value("sip.transport", default_transport);
    }
    writeLog("username = " + sip_username + "<br>password = " + sip_password + "<br>domain = " + sip_domain + "<br>transport = " + sip_transport + "<br>target = " + sip_target + "<br>);
    sipbutton.disabled = true;
    sipRegistered.className = "LED_off";
    countDown(0);
    sipStatus.innerHTML = "Connecting to server...";
    writeLog("Starting SIP daemon...");
    sipphone.start(sip_username, sip_password, sip_domain, sip_transport);
    writeLog("Connecting signals...");
    sipphone.ready.connect(onReady);
    writeLog("onReady() connected to sipphone.ready");
    sipphone.registered.connect(onRegistered);
    writeLog("onRegistered() connected to sipphone.registered");
    sipphone.placingCall.connect(onPlacingCall);
    writeLog("onPlacingCall() connected to sipphone.placingCall");
    sipphone.established.connect(onEstablished);
    writeLog("onEstablished() connected to sipphone.established");
    sipphone.disconnected.connect(onDisconnected);
    writeLog("onDisconnected() connected to sipphone.disconnected");
    sipphone.ring.connect(onRing);
    writeLog("onRing() connected to sipphone.ring");
    sipphone.incomingCall.connect(onIncomingCall);
    writeLog("onIncomingCall() connected to sipphone.incomingCall");
    sipphone.error.connect(onError);
    writeLog("onError() connected to sipphone.error");
    writeLog("SIP widget started, all signals are connected.");
}
function onReady(){
    writeLog("sipphone.status = " + sipphone.status);
    sipbutton.disabled = true;
    sipbutton.className = "callButton";
    sipStatus.innerHTML = "Call";
    sipTimer.className = "timerOff";
    //sipRegistered.className = "LED_white";
    writeLog('onReady() READY');
}
var checkRegistrationStatusTimeout;
function onRegistered()
    clearTimeout(checkRegistrationStatusTimeout);
    writeLog("sipphone.status = "+sipphone.status);
    var success = (sipphone.status=="register successful"); // CHECK REGISTRATION STATUS
    if(success){
        sipbutton.disabled = false;
        sipbutton.className="callButton";
        sipbutton.innerHTML="Call";
        sipStatus.innerHTML = "Ready";
        sipTimer.className = "timerOff";
        sipRegistered.className = "LED_green";
        writeLog('onRegistered() REGISTERED');
    } else {
        sipbutton.disabled = true;
        sipbutton.className="callButton";
        sipStatus.innerHTML = "\n        sipTimer.className = "timerOff";
        sipRegistered.className = "LED_off";
        writeLog('Waiting for server...');
        checkRegistrationStatusTimeout = setTimeout("onRegistered()", 15000);
    }
}

function onPlacingCall()
    writeLog("sipphone.status = "+sipphone.status);
    sipbutton.className="hangupButton";
    sipbutton.innerHTML="Cancel";
    sipStatus.innerHTML = "Placing call...";
    sipTimer.className = "timerOn";
    writeLog('onPlacingCall()');
}

function onIncomingCall()
    writeLog("sipphone.status = "+sipphone.status);
    sipStatus.innerHTML = "Incoming call";
    sipTimer.className = "timerOn";
    writeLog('onIncomingCall()');
}

function onEstablished()
    sipStatus.innerHTML = "In Call";
    sipTimer.className = "timerOn";
    callInProgress = true;
    countDown(1);
    writeLog('onEstablished()');
    writeLog("callInProgress = "+callInProgress);
}

function onRing()
    sipStatus.innerHTML = "Calling...";
    sipTimer.className = "timerOn";
    writeLog('onRing()');
}

function onDisconnected(){
writeLog("sipphone.status = "+sipphone.status);
callInProgress = false;
sipbutton.disabled = false;
sipbutton.className="callButton";
sipbutton.innerHTML="Call";
sipStatus.innerHTML = "Ready";
sipTimer.className = "timerOff";
countDown(0);
writeLog('onDisconnected()');
}
var t1;
function onError(code, explanation){
writeLog("sipphone.status = "+sipphone.status);
callInProgress = false;
sipbutton.disabled = true;
sipbutton.className="callButton";
sipbutton.innerHTML="Call";
sipTimer.className = "timerOff";
countDown(0);
switch(code){
case 404:
sipStatus.innerHTML = "<span style='color:#ff0000;'>No answer</span>";
break;
case 401:
sipStatus.innerHTML = "<span style='color:#ff0000;'>Registration failed</span>";
break;
default:
sipStatus.innerHTML = "<span style='color:#ff6920;'>Error</span>";
break;
}
t1 = setTimeout(function(){
sipbutton.disabled = false;
sipStatus.innerHTML = "Ready";
}, 30000);
writeLog("onError() " + explanation + " (SIP code = " + code +")");
}
function makeCall(targetID){
writeLog("sipphone.status = "+sipphone.status);
sipbutton.disabled = true;
var uri = targetID ? targetID : sip_target;
uri = uri.indexOf("sip:")<0 ? "sip:" + uri : uri;
if(callInProgress){
callInProgress = false;
sipphone.hangup();
writeLog("hangup(); callInProgress = "+callInProgress);
} else {
callInProgress = true;
sipphone.call(uri);
writeLog("calling " + uri" / callInProgress = "+callInProgress);
}
// For Timing to be shown
var sip_sec = 00; // set the seconds
var sip_min = 00; // set the minutes
var sip_hrs = 00; // set the Hours
var sip_OneSecond;
function countDown(flag){
var calltime;
if (flag) {
sip_sec++;
if (sip_sec == 59) {
 sip_sec = 00;
sip_min = sip_min + 1;
}
if (sip_min == 59) {
    sip_min = 00;
    sip_hrs = sip_hrs + 1;
}
if (sip_sec <= 9) {
    sip_sec = "0" + sip_sec;
}
calltime = ((sip_hrs<1 ? "0" : ((sip_hrs<=9 ? "0" : sip_hrs) + ":")) +
(sip_min<1 ? "0" : (sip_min == 9 & sip_hrs>0 ? "0" + sip_min : sip_min)) + ":" + sip_sec;
sipTimer.innerHTML = calltime;
sipTimer.title = "Last call duration: " + calltime;
sip_OneSecond = setTimeout("countDown(1)", 1000);
} else {
    sipTimer.innerHTML = "0:00";
    clearTimeout(sip_OneSecond);
    sip_sec = 00;
    sip_min = 00;
}
}
function isDebugMode() {
    var l = String(window.location);
    var qs = l.substring(l.indexOf("?", 0)+1, l.length);
    if (qs.indexOf("debug", 0)>=0) {
        document.getElementById('appDebugInfo').style.visibility = "visible";
    } else {
        document.getElementById('appDebugInfo').style.visibility = "hidden";
    }
}
function init() {
    isDebugMode();
    initSIP();
}</script>
</head>
<body onLoad="init()">
<div class="fullwindow vbox">
    <div class="vbox" style="-webkit-box-flex: 1; box-flex: 1; -webkit-box-pack: center; box-pack: center;">
        <table border="0" cellpadding="0" cellspacing="0" width="100%" height="100%">
            <tr>
                <td align="center">
                    <table border="0" cellpadding="0" cellspacing="1" align="center"
                        bgcolor="#000000">
                        <tr>
                            <td class="topPanel">
                                <table border="0" cellpadding="0" cellspacing="0" width="100%" height="100%" style="width:100%; max-width:640px">
                                    <tr>
                                        <td width="49%" style="min-width:160px" align="left"><span class="status" id="SIPstatus"></span></td>
                                        <td align="center">
                                            <div style="min-width:162px">
                                                <table border="0" cellpadding="0" cellspacing="0" height="100%">
                                                    <tr>
                                                        <td width="44"><div style="width:44px"></div></td>
                                                        <td width="74" align="center" style="width:74px"><img id="camera" src="img/eye.png" width="30" height="30" /></td>
                                                        <td width="44"><img id="registeredLED" class="LED_off" src="img/spacer.png" width="44" height="24" /></td>
                                                    </tr>
                                                </table>
                                            </div>
                                        </td>
                                        <td width="49%" style="min-width:160px" align="right">&nbsp;</td>
                                    </tr>
                                </table>
                            </td>
                        </tr>
                    </table>
                </td>
            </tr>
        </table>
    </div>
</div>
</body>
The above HTML code creates a sample widget to test SIP.

1. Copy the above contents to an HTML file and transfer the file to a web server from where it can be accessed. Make sure to have the location of the file as a URL.

2. Register the IEC and the other endpoint on CUCM (see instructions below)

3. Create a policy with startup URL as the URL of the above script. Make changes to the application->data property in the policy. Apply the policy. (see instructions below)

Note
The resolution of the video call from a Cisco video IP phone is fixed. It cannot be adjusted.

IEC Preparation

The following steps must be done before setting up the SIP client.

Step 1
Make sure that the IEC is installed, registered, configured, and up and running. Confirm that the startup URL is displaying.

Step 2
Connect a webcam using a USB cable to a USB port on the IEC.

Step 3
Connect a microphone to the IEC. You can connect the microphone to either a USB port or the MIC-in port.

SIP Client

In order for the SIP to work, the Cisco IEC 4650 device and Cisco IP Phone will need to be configured on the Cisco Unified Communications Manager (CUCM) and then configured on the Cisco IEM.

To install the SIP Client, you will need the following:

- CUCM version 9.x or 10.x
- Cisco IEC4650
- Cisco Unified IP Phone 9951
- Cisco TelePresence PrecisionHD USB Camera

Cisco IP Phone Set Up on the CUCM

The following steps will set up a Cisco Unified IP Phone 9951 on the CUCM. Modify the values entered if you are setting up a different phone, Tandberg, or TelePresence.
Step 1 Enter the IP address of your CUCM in your browser.
Step 2 Press the Enter button.
Step 3 In the CUCM main page, select Cisco Unified Communications Manager. You will be prompted to the Website’s Security Certificate page.
Step 4 On the Website’s Security Certificate page, click Continue to this website (Not Recommended).
Step 5 Enter admin in the Username field of the Cisco Unified CM Administration page.
Step 6 Enter the password in the Password field.
Step 7 Click Login button.
Step 8 From the Device drop-down menu, choose Phone.
Step 9 Click the Find button. All the devices registered on the CUCM will be listed.
Step 10 To add a new phone, click Add New.
Step 11 From the Phone Type drop-down menu, choose Cisco 9951.
Step 12 Click Next.
Step 13 Enter the IP phone’s MAC address in the MAC Address field within the Device Information area.

---

**Note**  If you do not know the IP phone’s MAC address, refer to the section “Finding the IP Phone’s MAC Address” at the end of this section.

---

Step 14 Enter a description of the IP phone to easily distinguish it from others in the CUCM. This field automatically enters the IP phone’s MAC Address but can be modified.
Step 15 From the Device Pool drop-down menu, choose Default.
Step 16 From the Phone Button Template drop-down menu, choose Standard 9951 SIP.
Step 17 From the Device Security Profile drop-down menu within the Protocol Specific Information area, choose Cisco 9951 - Standard SIP Non-Secure Profile.
Step 18 From the SIP Profile drop-down menu, choose Standard SIP Profile.
Step 19 Within the Protocol Specific Information area, go to the Digest User drop-down menu and choose the User ID.
Step 20 From the Cisco Camera drop-down menu within the Product Specific Configuration Layout, choose Enabled.
Step 21 From the Video Capabilities drop-down menu, choose Enabled.
Step 22 From the Web Access drop-down menu, choose Enabled.
Step 23 Click Save. A dialog box appears.
Step 24 Click Apply Config.

---

**Note** It is important that you first save configurations before applying them. Otherwise, the configurations will be lost.

---

Step 25 Click OK.
Step 26  Click **Line [1] – Add a new DN** within the Association Information area.

Step 27  Enter the directory number in the Directory Number field. The directory number must be a number that does not already exist in the CUCM.

Step 28  Enter a description in the Description field. It is good practice to enter the directory number in this field.

Step 29  Enter a value in the Alerting Name field. It is good practice to enter the directory number in this field too.

Step 30  Enter a description in the ASCII Alerting Name field. It is good practice to enter the directory number in this field too.

Step 31  Click **Save**.

Now that a directory number has been specified, the IP phone must be configured to pick this number and store it. To do so, it has to be linked to the CUCM server.

Step 32  Go to the IP phone.

Step 33  Press the System Settings button.

Step 34  Choose the **Administrator Settings** icon, which is button #4 on the Applications screen.

Step 35  Choose the Network Setup icon, which is button #1 on the Administrator Settings screen.

Step 36  Choose the Ethernet Setup icon, which is button #1 on the Network Setup screen.

Step 37  Choose the IPv4 Setup icon, which is button #1 on the Ethernet Setup screen.

Step 38  Choose the Alternative TFTP icon, which is button #8 on the IPv4 Setup screen.

Step 39  In the TFTP Server 1 field, enter the IP Address of the CUCM Server.

---

**Finding The IP Phone’s MAC Address**

The Cisco Unified IP phone 9951 has a MAC address, which can be found by one of two methods.

**First Method**

There is a label on the bottom of the phone that contains the MAC address.

**Second Method**

Step 1  Press the System Settings button.

Step 2  Choose the **Administrator Settings** icon, which is button #4 on the Applications screen.

Tip  You can either use the touch screen on the display or the numbers on the keypad to navigate the phone settings.

Step 3  Choose the Network Setup icon, which is button #1 on the Administrator Settings screen.

Step 4  Choose the Ethernet Setup icon, which is button #1 on the Network Setup screen.

Step 5  Choose the MAC Address icon, which is button #2 on the Ethernet Setup screen.
Cisco IEC Set Up on the CUCM

The Cisco IEC 4650 device set up on the CUCM is very similar to the Cisco IP Phone 9951 set up on the CUCM except for a few options. An additional step is also required. This step is the setting up of a User Profile. The User Profile is then linked to the Cisco IEC 4650 device after it is set up on the CUCM.

Step 1: Enter the IP address of your CUCM in your browser.
Step 2: Press the Enter button.
Step 3: In the CUCM main page, select Cisco Unified Communications Manager.
You will be prompted to the Website’s Security Certificate page.
Step 4: On the Website’s Security Certificate page, click Continue to this website (Not Recommended).
Step 5: Enter admin in the Username field of the Cisco Unified CM Administration page.
Step 6: Enter the password in the Password field.
Step 7: Click Login button.
Step 8: From the Device drop-down menu, choose Phone.
Step 9: Click the Find button.
All the devices registered on the CUCM will be listed.
Step 10: To add a new phone, click Add New.
Step 11: From the Phone Type drop-down menu, choose Third Party SIP Device (Advanced).
Step 12: Click Next.
Step 13: Enter the Cisco IEC 4650 device’s MAC address in the MAC Address field within the Device Information area.

Note: The Cisco IEC 4650 device’s MAC address is located on the label on the back of the device.

Step 14: Enter a description of the Cisco IEC 4650 device. This field automatically enters “SEP” plus the MAC Address but the field can be modified.
Step 15: From the Device Pool drop-down menu, choose Default.
Step 16: From the Phone Button Template drop-down menu, choose Third Party SIP Device (Advanced).
Step 17: From the SIP Profile drop-down menu, choose Standard SIP Profile.
Step 19: Click Save.
Step 20: Click Apply Config.
In order for the IEC 4650 device to be activated, it must be associated with a User Profile.
Step 21: From the User Management drop-down menu, choose End User.
Step 22: Click Add New.
Step 23: Enter a value in the User ID field. A unique numeric value is required to identify the user. This unique value will be the extension of the SIP device.
Note: It is imperative that the value entered in the User ID field is a number. The SIP device will not work if you enter alphabetic characters, punctuation, or spaces.

Step 24 Enter a password in the Password field.
Step 25 Re-enter the password in the Confirm Password field.
Step 26 Enter the last name of the user in the Last Name field.
Step 27 Click Save.
   You will be redirected to a page where you can find the status of your User Profile creation. If all fields have been entered properly the status will indicate ‘Add Successful’.
   The user profile and the Cisco IEC 4650 device on the CUCM must now be linked in order for the phone to work.
Step 28 From the Device drop-down menu, choose Phone.
Step 29 Click the Find button.
   All the devices registered on the CUCM will be listed.
Step 30 Choose the Cisco IEC 4650 device, which starts with the letters “SEP” followed by the MAC address.
Step 31 On the Phone Configuration screen, choose Line [1] – Add a new DN within the Association Information area.
   The Directory Number Configuration page appears.
Step 32 Enter a number in the Directory Number field.
Step 33 Click Save.
Step 34 Click Associate End Users.
   The user list screen appears.
Step 35 Click Find.
Step 36 Check the check box next to the user that you would like to associate the IEC directory number.
Step 37 Click Add Selected.
Step 38 Click Close.
Step 39 From the Device drop-down menu, choose Phone.
Step 40 Click Save.
Step 41 Click Apply Config.
Step 42 Within the Protocol Specific Information area, go to the Digest User drop-down menu and choose the User ID.
Step 43 Click Save.
Step 44 Click Apply Config.
   This Cisco IEC 4650 device is now registered on the CUCM.
Configuring Call Manager Information

Once the end points (the video IP phone and an IEC) have been registered on the CUCM, you have several options for configuring the call manager information so that the IEC can call or receive calls from the video IP phone:

- You can enter the call manager information in a policy on the IEM.
- You can hard code the call manager information in the sipphone widget.

Using a Policy on the IEM

The following steps explain how to enter the call manager information into the IEC’s policy on the IEM.

Step 1 Log in to the Cisco IEM which has the SIP policy enabled on it.
Step 2 Go to the policy that is applied to the Cisco IEC4610 or 4632 device.
Step 3 Click the Policy tab.
Step 4 Expand the application property.
Step 5 In the data property, click the value field.
Step 6 In the Application data editor, click +.
Step 7 Click key:value.
Step 8 Enter sip.target in the key field.
Step 9 Enter the directory number in the value field.
Step 10 Click Ok.

If you click on data property’s Value field, you will see the data in the form sip.target:[directoryNumber].

Next you will add the username, password, domain, and transport protocol for the Cisco Unified Communications Manager (CUCM).

Step 11 In the Application data editor, click +.
Step 12 Enter sip.username in the key field.
Step 13 In the value field, enter the username that the IEM will use to log into the CUCM. This is the unique User ID that was entered in the CUCM.

Note It is imperative that the value entered in the User ID field is a number. The SIP device will not work if you enter alphabetic characters, punctuation, or spaces.

Step 14 In the Application data editor, click +.
Step 15 Enter sip.password in the key field.
Step 16 In the value field, enter the password that the IEM will use to log into the CUCM.
Step 17 In the Application data editor, click +.
Step 18 Enter sip.domain in the key field.
Step 19 In the value field, enter the IP address of the CUCM.
Step 20 In the Application data editor, click +.
Step 21 Enter `sip.transport` in the key field.
Step 22 Enter `udp` in the value field.

**Note**
It is important to enter all values in lowercase characters. If you enter “UDP” instead of “udp”, the call will not work.

Step 23 Click Ok.
If you click on the data property’s Value field, you will see the data.

Step 24 Click Apply.

---

**Using the SipPhone Widget**

You can hard code the call manager information in the sipphone widget. If you want to hard code the SIP client information in this widget, follow these steps.

Step 1 Open the sipphone widget code using a text editor.
Step 2 Find the `sipphone.start(username, password, domain, transport)` line in the HTML as shown below.

```html
....
{
    sipphone = document.getElementById("sipphone");

    // Now Call Start Routine with the SIP Credentials
    // that we got from the applicationData
    sipphone.start(username, password, domain, transport);

    sipphone.placingCall.connect(onPlacingCall);
    sipphone.incomingCall.connect(onIncomingCall);
    sipphone.ready.connect(onReady);
    sipphone.registered.connect(onRegistered);
    sipphone.established.connect(onEstablished);
    sipphone.ring.connect(onRing);
    sipphone.disconnected.connect(onDisconnected);
    sipphone.error.connect(onError);
}
....
```

Step 3 Replace “username” with the call manager ID, which is a number.
Step 4 Replace “password” with call manager’s password.
Step 5 Replace “domain” with the IP address for the call manager.
Step 6 Replace “transport” with “udp”.
Step 7 Save your changes.
SIP DTMF

Dual-Tone Multifrequency (DTMF) for SIP is a feature that is available starting with version 2.1.1. The purpose of DTMF setup for SIP is to provide the audio prompts heard over the phone such as “Press 1 to reach ____.”

You will need the following to set up DTMF for SIP calls:

- Latest SipPhone widget with the sendDtmf line of code.
  ```java
  void sendDtmf(in string dtmfkey);
  ```
- Cisco Unified Contact Center Express (UCCX) 9.x: UCCX provides DTMF capability.
- CUCM 9.x: Although CUCM does not have the DTMF feature, when configuring the IEC as a SIP device in the CUCM, the following options must be configured in order for DTMF to work correctly:
  1. Follow the steps in the following link to enable the Media Termination Point system wide for the CUCM: [http://www.cisco.com/en/US/docs/voice_ip_comm/cucm/admin/3_0_9/p4mtp.html](http://www.cisco.com/en/US/docs/voice_ip_comm/cucm/admin/3_0_9/p4mtp.html)
  2. The IEC checkbox should be unchecked.
  3. The Unattended Port checkbox should be unchecked.
  4. The Require DTMF Reception checkbox should be unchecked.
  5. The Allow Presentation Sharing using BFCP checkbox should be unchecked.
  6. The Allow iX Applicable Media checkbox should be unchecked.

Sample usage of sendDtmf() API

This section provides an example on how to use the sendDtmf() API.

The following is the Javascript Function to send the DTMF keys:

```javascript
function sendDtmf(key){
    var k = String(key);
    var validValues = "0123456789*#";
    if(validValues.indexOf(k)>=0){
        writeLog("sendDtmf('"+k+'\n')
        sipphone.sendDtmf(k);
        writeLog('ok');
    } else {
        writeLog('Invalid DTMF argument.');
    }
}
```

The HTML code to bind the keys to the function is:

```html
<table>
  <tr>
    <td><button onClick="sendDtmf('1')" class="siphone_key">1</button></td>
    <td><button onClick="sendDtmf('2')" class="siphone_key">2</button></td>
    <td><button onClick="sendDtmf('3')" class="siphone_key">3</button></td>
    <!-- More such lines for each of the DTMF keys -->
  </tr>
</table>
```