

Bandwidth Constraints

The screenshot shows a browser window with the URL <http://ef21-77-226-74-230.ngrok.io/bandwidthconstraints/>. The page title is "Bandwidth Constraints". On the left, there is a code editor with Twilio-video JavaScript code. In the center, there is a video preview of a participant wearing glasses and a grey shirt. Below the video is a chart titled "Select Bandwidth Constraints" showing "Max Audio Bitrate (bps)" at 32000 and "Max Video Bitrate (bps)" at 128000. To the right, the browser's developer tools Console tab shows a log of TwilioConnection events, including heartbeat messages and connection status updates.

```
'use strict';

var Video = require('twilio-video');

/**
 * Connect to a Room with the given bandwidth constraints.
 * @param {string} token - Token for joining the Room
 * @param {string} roomName - Room name
 * @param {number} maxAudioBitrate - Max audio bitrate (bps)
 * @param {number} maxVideoBitrate - Max video bitrate (bps)
 * @returns {Promise<Room>}
 */
function connectWithBandwidthConstraints(token, roomName, maxAudioBitrate, maxVideoBitrate) {
    return Video.connect(token, {
        maxAudioBitrate: maxAudioBitrate,
        maxVideoBitrate: maxVideoBitrate,
        name: roomName
    });
}

/**
 * Update the bandwidth constraints of a Room.
 * @param {Room} room - The Room whose bandwidth constraints have to be updated
 * @param {number} maxAudioBitrate - Max audio bitrate (bps)
 * @param {number} maxVideoBitrate - Max video bitrate (bps)
 */
function updateBandwidthConstraints(room, maxAudioBitrate, maxVideoBitrate) {
    room.localParticipant.setParameters({
        maxAudioBitrate: maxAudioBitrate,
        maxVideoBitrate: maxVideoBitrate
    });
}

exports.connectWithBandwidthConstraints = connectWithBandwidthConstraints;
exports.updateBandwidthConstraints = updateBandwidthConstraints;
```

Room Sid: RMcf555ab48c96f563ede60f80283da4ee

Local Video Snapshot

The screenshot shows a browser window with the URL <http://ef21-77-226-74-230.ngrok.io/localvideosnapshot/>. The page title is "Local Video Snapshot". On the left, there is a code editor with Twilio-video JavaScript code. In the center, there is a video preview of a participant wearing glasses and a grey shirt. Below the video is a blue button labeled "Take Snapshot". To the right, the browser's developer tools Console tab shows a log of TwilioConnection events, including media stream track creation and local video track initialization.

```
'use strict';

var Video = require('twilio-video');

/**
 * Display local video in the given HTMLVideoElement.
 * @param {HTMLVideoElement} video
 * @returns {Promise<void>}
 */
function displayLocalVideo(video) {
    return Video.createLocalVideoTrack().then(function(localTrack) {
        localTrack.attach(video);
        return localTrack;
    });
}

/**
 * Take snapshot of the local video from the HTMLVideoElement and render it in the HTMLCanvasElement.
 * @param {HTMLVideoElement} video
 * @param {LocalVideoTrack} localVideoTrack
 * @param {HTMLCanvasElement|HTMLImageElement} snapshot
 */
function takeLocalVideoSnapshot(video, localVideoTrack, snapshot) {
    if (window.ImageCapture) {
        const imageCapture = new ImageCapture(localVideoTrack.mediaStreamTrack);
        imageCapture.takePhoto().then(function(blob) {
            snapshot.src = URL.createObjectURL(blob);
        });
    } else {
        snapshot.getContext('2d').drawImage(video, 0, 0);
    }
}

module.exports.displayLocalVideo = displayLocalVideo;
module.exports.takeLocalVideoSnapshot = takeLocalVideoSnapshot;
```

Room Sid: RMcf555ab48c96f563ede60f80283da4ee

Codec Preferences

The screenshot shows a browser window with two main tabs: "Codec Preferences" and "Participant's Media".

- Codec Preferences Tab:** Displays a code editor with a snippet of JavaScript. The code defines a function `connectWithPreferredCodecs` that takes parameters like `token`, `roomName`, and `preferredAudioCodecs`. It uses the `Video` API to connect and sets the preferred audio and video codecs.
- Participant's Media Tab:** Shows a video feed of a person wearing glasses and a grey shirt.
- Developer Tools Console:** Located at the bottom of the browser window, it displays a log of network requests and events. The logs include entries for Twilio connections, signaling, and heartbeats, with timestamps ranging from 2021-11-26T15:50 to 2021-11-26T15:51.

Room Sid: RM6ba364e850efb438d701cf2bd2aa572a

Dominant Speaker Detection

Dominant Speaker Detection

```
'use strict';

var Video = require('twilio-video');

/**
 * Connect to a Room with the Dominant Speaker API enabled.
 * This API is available only in Small Group or Group Rooms.
 * @param {string} token - Token for joining the Room
 * @returns {CancelablePromise<Room>}
 */
function connectToRoomWithDominantSpeaker(token) {
    return Video.connect(token, {
        dominantSpeaker: true
    });
}

/**
 * Listen to changes in the dominant speaker and update your application.
 * @param {Room} room - The Room you just joined
 * @param {function} updateDominantSpeaker - Updates the app UI with the new dominant speaker
 * @returns {void}
 */
function setupDominantSpeakerUpdates(room, updateDominantSpeaker) {
    room.on('dominantSpeakerChanged', function(participant) {
        console.log(`A new RemoteParticipant is now the dominant speaker!`); // participant is the new dominant speaker
        updateDominantSpeaker(participant);
    });
}

exports.connectToRoomWithDominantSpeaker = connectToRoomWithDominantSpeaker;
exports.setupDominantSpeakerUpdates = setupDominantSpeakerUpdates;
```

Join Room:

RMbc56eebe4ba556881a6de9b8b801de95

Please join the above Room using the [QuickStart App](#) or connect the below Participants to the Room and unmute only one of them.

Alice	Mak
<input style="width: 100%; height: 30px; border: 1px solid #ccc; border-radius: 5px; padding: 5px; margin-bottom: 5px;" type="button" value="Disconnect"/> <input style="width: 100%; height: 30px; border: 1px solid #ccc; border-radius: 5px; padding: 5px;" type="button" value="Mute"/>	<input style="width: 100%; height: 30px; border: 1px solid #ccc; border-radius: 5px; padding: 5px; margin-bottom: 5px;" type="button" value="Disconnect"/> <input style="width: 100%; height: 30px; border: 1px solid #ccc; border-radius: 5px; padding: 5px;" type="button" value="Unmute"/>
Bob	Charlie
<input style="width: 100%; height: 30px; border: 1px solid #ccc; border-radius: 5px; padding: 5px; margin-bottom: 5px;" type="button" value="Disconnect"/> <input style="width: 100%; height: 30px; border: 1px solid #ccc; border-radius: 5px; padding: 5px;" type="button" value="Unmute"/>	<input style="width: 100%; height: 30px; border: 1px solid #ccc; border-radius: 5px; padding: 5px; margin-bottom: 5px;" type="button" value="Disconnect"/> <input style="width: 100%; height: 30px; border: 1px solid #ccc; border-radius: 5px; padding: 5px;" type="button" value="Unmute"/>

2021-11-26T15:53:01.155Z debug [TwilioConnection #4 ws_index#s:17772 /@/global.vss.twilio.com/signaling] Incoming: {"type": "heartbeat"}

2021-11-26T15:53:03.689Z debug [TwilioConnection #2 ws_index#s:17772 /@/global.vss.twilio.com/signaling] Incoming: {"type": "heartbeat"}

2021-11-26T15:53:03.760Z debug [TwilioConnection #3 ws_index#s:17772 /@/global.vss.twilio.com/signaling] Incoming: {"type": "heartbeat"}

2021-11-26T15:53:03.763Z debug [TwilioConnection #1 ws_index#s:17772 /@/global.vss.twilio.com/signaling] Outgoing: {"type": "heartbeat"}

2021-11-26T15:53:03.796Z debug [TwilioConnection #3 ws_index#s:17772 /@/global.vss.twilio.com/signaling] Incoming: {"type": "heartbeat"}

2021-11-26T15:53:03.800Z debug [TwilioConnection #4 ws_index#s:17772 /@/global.vss.twilio.com/signaling] Outgoing: {"type": "heartbeat"}

2021-11-26T15:53:03.995Z info [elapsedTime: 10682, group: 'quality', level: 'info', name: 'stats-reporter']

2021-11-26T15:53:04.000Z info [elapsedTime: 10682, group: 'connect']

2021-11-26T15:53:04.000Z debug [TwilioConnection #1 ws_index#s:17772 /@/global.vss.twilio.com/signaling] event: {"type": "ice-candidate-pair", "timestamp": 163794198995, "j

2021-11-26T15:53:05.217Z debug [TwilioConnection #3 ws_index#s:17772 /@/global.vss.twilio.com/signaling] Incoming: {"type": "heartbeat"}

2021-11-26T15:53:06.249Z debug [TwilioConnection #1 ws_index#s:17772 /@/global.vss.twilio.com/signaling] Incoming: {"type": "heartbeat"}

2021-11-26T15:53:06.255Z debug [TwilioConnection #4 ws_index#s:17772 /@/global.vss.twilio.com/signaling] Incoming: {"type": "heartbeat"}

2021-11-26T15:53:06.332Z debug [TwilioConnection #3 ws_index#s:17772 /@/global.vss.twilio.com/signaling] Incoming: {"type": "heartbeat"}

2021-11-26T15:53:06.846Z debug [TwilioConnection #5 ws_index#s:17772 /@/global.vss.twilio.com/signaling] Outgoing: {"type": "heartbeat"}

2021-11-26T15:53:06.887Z info [elapsedTime: 1, group: 'quality', level: 'info', name: 'stats-reporter']

2021-11-26T15:53:06.888Z info [elapsedTime: 1, group: 'connect']

2021-11-26T15:53:06.891Z info [elapsedTime: 1, group: 'active-ice-candidate-pair']

2021-11-26T15:53:06.913Z info [elapsedTime: 30615, group: 'quality', level: 'info', name: 'stats-reporter']

2021-11-26T15:53:06.913Z info [elapsedTime: 30615, group: 'connect']

2021-11-26T15:53:07.078Z debug [TwilioConnection #2 ws_index#s:17772 /@/global.vss.twilio.com/signaling] Outgoing: {"type": "heartbeat"}

2021-11-26T15:53:07.078Z debug [TwilioConnection #4 ws_index#s:17772 /@/global.vss.twilio.com/signaling] Outgoing: {"type": "heartbeat"}

2021-11-26T15:53:08.564Z debug [TwilioConnection #1 ws_index#s:17772 /@/global.vss.twilio.com/signaling] Outgoing: {"type": "heartbeat"}

2021-11-26T15:53:08.566Z debug [TwilioConnection #3 ws_index#s:17772 /@/global.vss.twilio.com/signaling] Outgoing: {"type": "heartbeat"}

2021-11-26T15:53:08.600Z debug [TwilioConnection #5 ws_index#s:17772 /@/global.vss.twilio.com/signaling] Outgoing: {"type": "heartbeat"}

Room Sid: RMbc56eebe4ba556881a6de9b8b801de95

Network Quality

The screenshot shows a browser window with two main sections: "Set Network Quality Verbosity Levels" and a "Console" tab displaying a log of network activity.

Set Network Quality Verbosity Levels

- LocalParticipant:** Set to 2.
- RemoteParticipant(s):** Set to 2.
- Leave Room:** A blue button.

Join Room: A text input field containing "RM58583c3faabc93238426d550efba4521".

After creating the Room with the desired verbosity levels, please join the above Room using the [QuickStart App](#) or connect the below Participants to the Room. You can also change the verbosity levels and observe the change in the verbosity of the reported Network Quality stats.

Participants:

- Charlie (Connected)
- Alice (Connected)
- Mak (Connected)
- Bob (Connected)

Network Quality Status:

NQ Level (You): 5

Console Log:

```
(elapsedTime: 121225, group: 'quality', level: 'info', name: 'stats-repo', rt, timestamp: 1637942186395, _)
2021-11-26T15:56:26.765Z debug [TwilioConnection #5: ws://global.vss.twilio.com/signaling] Incoming: {"type": "heartbeat"}
2021-11-26T15:56:26.837Z debug [TwilioConnection #2: ws://global.vss.twilio.com/signaling] Outgoing: {"type": "heartbeat"}
2021-11-26T15:56:28.082Z debug [TwilioConnection #1: ws://global.vss.twilio.com/signaling] Incoming: {"type": "heartbeat"}
2021-11-26T15:56:28.208Z debug [TwilioConnection #2: ws://global.vss.twilio.com/signaling] Outgoing: {"type": "heartbeat"}
2021-11-26T15:56:28.228Z debug [TwilioConnection #2: ws://global.vss.twilio.com/signaling] Incoming: {"type": "heartbeat"}
2021-11-26T15:56:28.239Z debug [TwilioConnection #2: ws://global.vss.twilio.com/signaling] Outgoing: {"type": "heartbeat"}
2021-11-26T15:56:28.259Z debug [TwilioConnection #4: ws://global.vss.twilio.com/signaling] Incoming: {"type": "heartbeat"}
2021-11-26T15:56:28.260Z debug [TwilioConnection #4: ws://global.vss.twilio.com/signaling] Outgoing: {"type": "heartbeat"}
2021-11-26T15:56:28.457Z debug [TwilioConnection #2: ws://global.vss.twilio.com/signaling] Incoming: {"type": "heartbeat"}
2021-11-26T15:56:28.457Z debug [TwilioConnection #2: ws://global.vss.twilio.com/signaling] Outgoing: {"type": "heartbeat"}
2021-11-26T15:56:29.418Z debug [TwilioConnection #3: ws://global.vss.twilio.com/signaling] Incoming: {"type": "heartbeat"}
2021-11-26T15:56:29.437Z debug [TwilioConnection #4: ws://global.vss.twilio.com/signaling] Incoming: {"type": "heartbeat"}
2021-11-26T15:56:30.304Z info [connect #2] event, index:js:27868
  * (elapsedTime: 124599, group: 'quality', level: 'info', name: 'stats-repo', rt, timestamp: 1637942191359, _)
2021-11-26T15:56:30.864Z debug [TwilioConnection #5: ws://global.vss.twilio.com/signaling] Incoming: {"type": "heartbeat"}
2021-11-26T15:56:31.399Z info [connect #3] event, index:js:27868
  * (elapsedTime: 1637942191359, group: 'quality', level: 'info', name: 'stats-repo', rt, timestamp: 1637942191359, _)
2021-11-26T15:56:31.807Z info [connect #4] event, index:js:27868
  * (elapsedTime: 126091, group: 'quality', level: 'info', name: 'stats-repo', rt, timestamp: 1637942191359, _)
2021-11-26T15:56:32.052Z debug [TwilioConnection #1: ws://global.vss.twilio.com/signaling] Incoming: {"type": "heartbeat"}
2021-11-26T15:56:32.526Z debug [TwilioConnection #2: ws://global.vss.twilio.com/signaling] Incoming: {"type": "heartbeat"}
2021-11-26T15:56:32.858Z info [connect #5] event, index:js:27868
  * (elapsedTime: 1637942192958, group: 'quality', level: 'info', name: 'stats-repo', rt, timestamp: 1637942192958, _)
2021-11-26T15:56:33.009Z debug [TwilioConnection #1: ws://global.vss.twilio.com/signaling] Incoming: {"type": "heartbeat"}
2021-11-26T15:56:33.052Z debug [TwilioConnection #2: ws://global.vss.twilio.com/signaling] Outgoing: {"type": "heartbeat"}
2021-11-26T15:56:33.079Z debug [TwilioConnection #3: ws://global.vss.twilio.com/signaling] Outgoing: {"type": "heartbeat"}
2021-11-26T15:56:33.092Z debug [TwilioConnection #3: ws://global.vss.twilio.com/signaling] Incoming: {"type": "heartbeat"}
2021-11-26T15:56:33.107Z debug [TwilioConnection #4: ws://global.vss.twilio.com/signaling] Outgoing: {"type": "heartbeat"}
2021-11-26T15:56:33.107Z debug [TwilioConnection #4: ws://global.vss.twilio.com/signaling] Incoming: {"type": "heartbeat"}
2021-11-26T15:56:33.051Z debug [TwilioConnection #5: ws://global.vss.twilio.com/signaling] Outgoing: {"type": "heartbeat"}
2021-11-26T15:56:33.519Z debug [TwilioConnection #3: ws://global.vss.twilio.com/signaling] Incoming: {"type": "heartbeat"}
2021-11-26T15:56:33.397Z debug [TwilioConnection #4: ws://global.vss.twilio.com/signaling] Outgoing: {"type": "heartbeat"}
2021-11-26T15:56:33.407Z debug [TwilioConnection #4: ws://global.vss.twilio.com/signaling] Incoming: {"type": "heartbeat"}
2021-11-26T15:56:34.963Z debug [TwilioConnection #5: ws://global.vss.twilio.com/signaling] Incoming: {"type": "heartbeat"}  
}
```

Room Sid: RM58583c3faabc93238426d550efba4521

Remote Participant Reconnection States

Room Sid: RM1cd4cf49505c9b114e8401b4631f1257

Video Track Manual Controls

Video Track Manual Controls

```
'use strict';

var Video = require('twilio-video');

/**
 * Connect to a Room with 'manual' mode. The default mode is 'auto'.
 * @param {string} token - AccessToken for joining the Room
 * @returns {Room}
 */
function joinRoom(token) {
  return Video.connect(token, {
    name: 'my-cool-room',
    bandwidthProfile: {
      video: {
        contentPreferencesMode: 'manual',
        clientTrackSwitchOffControl: 'manual'
      }
    }
  });
}

/**
 * Switch on the RemoteVideoTrack.
 * @param {RemoteVideoTrack} track - The RemoteVideoTrack you want to switch on
 * @returns {RemoteVideoTrack}
 */
function switchOn(track) {
  return track.switchOn();
}

/**
 * Switch off the RemoteVideoTrack.
 * @param {RemoteVideoTrack} track - The RemoteVideoTrack you want to switch off
 * @returns {RemoteVideoTrack}
 */
function switchOff(track) {
  return track.switchOff();
}

/**
 * Set the render dimensions of the RemoteVideoTrack.
 * @param {RemoteVideoTrack} track - The RemoteVideoTrack you want to set dimensions for
 * @param {{height: number, width: number}} renderDimensions - The render dimensions
 * @returns {RemoteVideoTrack}
 */
function setContentPreferences(dimensions) {
  return new Promise((resolve, reject) => {
    const videoTrack = document.querySelector('video');
    if (!videoTrack) {
      reject('Video track not found');
    } else {
      videoTrack.setDimensions(dimensions);
      resolve();
    }
  });
}
```

Remote Video Controls

Switch Off Control:

Content Preferences:

Remote Video Track



On

Video Bitrate



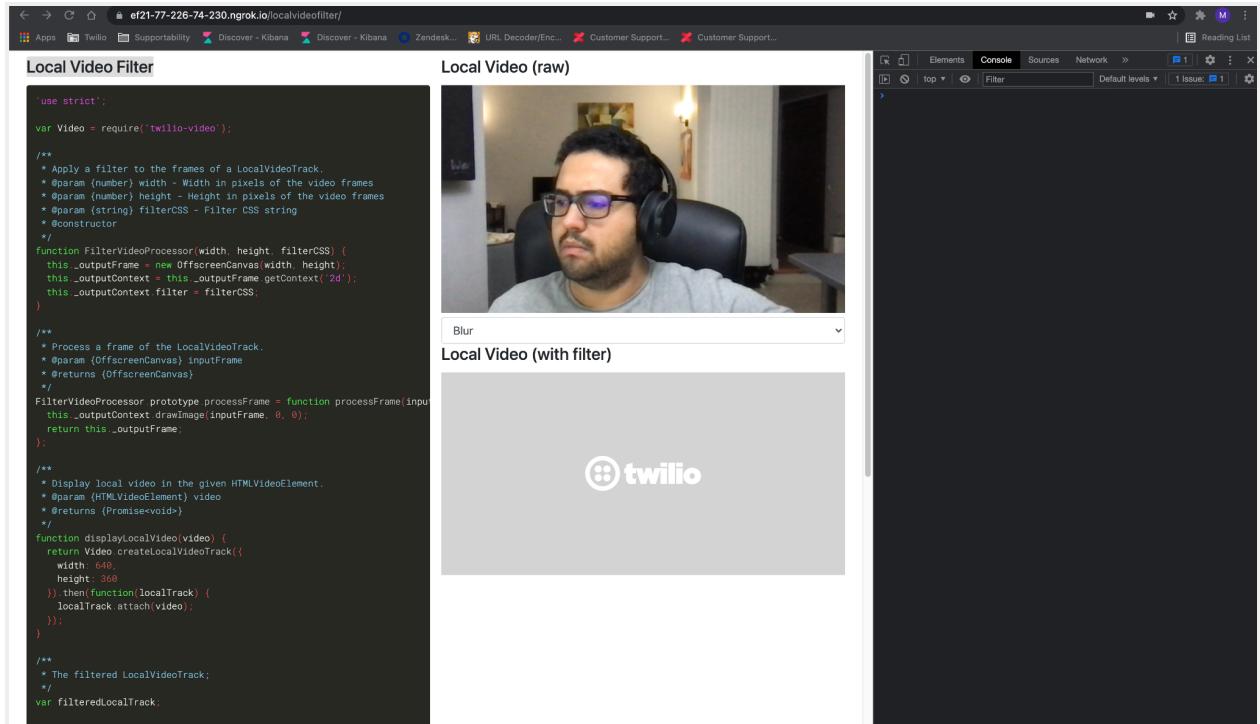
Reading List

Console

- Uncaught TypeError: Cannot read properties of null (reading 'index.js:29')
- switchOn()
 at HTMLButtonElement.switchOnBtn.onclick (index.js:147)
- Uncaught TypeError: Cannot read properties of null (reading 'index.js:48')
 at setContentPreferences()
 at setRenderDimensions (index.js:148)
 at HTMLSelectElement.<anonymous> (index.js:164)

Room Sid: RM55f6e5e406c6aadd7248d3867bc70181

Local Video Filter



Room Sid: RM55f6e5e406c6aadd7248d3867bc70181

Media Device Selection

The screenshot shows a web browser window with the URL <http://ef21-77-226-74-230.ngrok.io/mediadevices/>. The page is titled "Media Device Selection". On the left, there is a code editor displaying JavaScript code for media device selection. On the right, there are three main sections: "Preview Media" showing a video feed of a man with glasses and a beard wearing headphones; "Select Media Devices" with dropdown menus for Audio Input (set to "MacBook Pro Microphone (Built-in)"), Video Input (set to "FaceTime HD Camera (Built-in) (05ac:8514)"), and Audio Output (set to "MacBook Pro Speakers (Built-in)"); and a "Join room:" section with a room ID "RMfdf1e091f0cea001d821f82336b880da" and a note about joining from another device/browser using the QuickStart App.

```
/* eslint-disable no-console */
'use strict';

var Video = require('twilio-video');

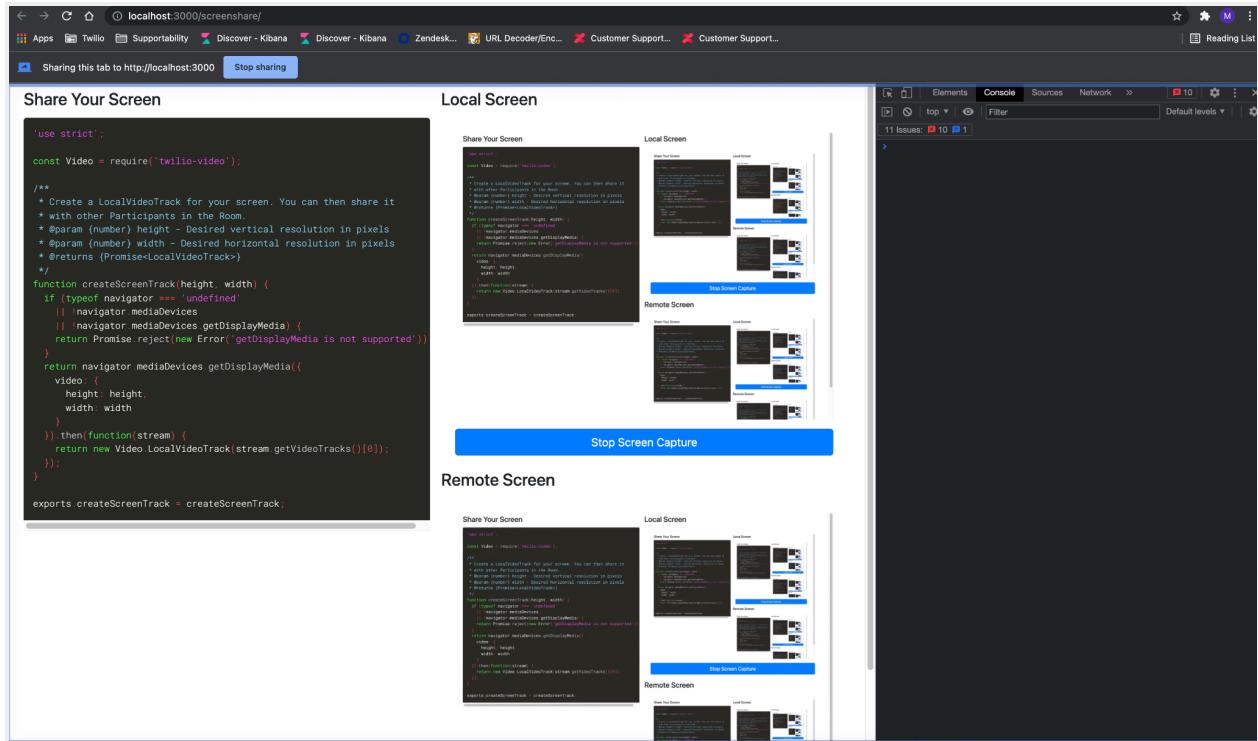
/**
 * Get the list of available media devices of the given kind.
 * @param {Array<MediaDeviceInfo>} deviceInfos
 * @param {string} kind - One of 'audioworklet', 'audioinput', 'audiooutput', 'videoinput'
 * @returns {Array<MediaDeviceInfo>} - Only those media devices of the given kind
 */
function getDevicesOfKind(deviceInfos, kind) {
  return deviceInfos.filter(function(deviceInfo) {
    return deviceInfo.kind === kind;
  });
}

/**
 * Apply the selected audio output device.
 * @param {string} deviceId
 * @param {HTMLAudioElement} audio
 * @returns {Promise<void>}
 */
function applyAudioOutputDeviceSelection(deviceId, audio) {
  return typeof audio.setSinkId === 'function'
    ? audio.setSinkId(deviceId)
    : Promise.reject('This browser does not support setting an audio output');
}

/**
 * Apply the selected input device.
 * @param {string} deviceId
 * @param {LocalTrack} localTrack - LocalAudioTrack or LocalVideoTrack
 * @param {'audio' | 'video'} kind
 * @returns {Promise<LocalTrack>} - The created or restarted LocalTrack
 */
function applyInputDeviceSelection(deviceId, localTrack, kind) {
  var constraints = { deviceId: { exact: deviceId } };
  if (localTrack) {
    return localTrack.restart(constraints).then(function() {
      return localTrack;
    });
  }
  return kind === 'audio'
    ? Video.createLocalAudioTrack(constraints)
    : Video.createLocalVideoTrack(constraints);
}
```

Room Sid: RMfdf1e091f0cea001d821f82336b880da

Share Your Screen



Room Sid: RM0f2070472aca0078abea3f2b970afb15

Reconnection States and Events

The screenshot shows a browser window at `localhost:3000/reconnection/`. On the left, a code editor displays a JavaScript file for handling reconnection events. In the center, a video player shows a man wearing glasses and headphones. Above the video, a message says "Room state is:" followed by three buttons: "Connected" (green), "Disconnected" (grey), and "Reconnecting" (white). On the right, the developer tools' console tab shows log messages related to media connection issues and reconnection attempts.

```
use strict';

/** 
 * Listen to Room reconnection events and update the UI accordingly.
 * @param {Room} room - The room you have joined
 * @param {function} updateRoomState - Updates the app UI with the new Room state
 * @returns {void}
 */
function setupReconnectionUpdates(room, updateRoomState) {
  room.on('disconnected', (room, error) => {
    if (error.code === 20104) {
      console.log(`Signaling reconnection failed due to expired AccessToken!`);
    } else if (error.code === 53800) {
      console.log(`Signaling reconnection attempts exhausted!`);
    } else if (error.code === 53204) {
      console.log(`Signaling reconnection took too long!`);
    }
    updateRoomState(room.state);
  });

  room.on('reconnected', () => {
    console.log('Reconnected to the Room!');
    updateRoomState(room.state);
  });

  room.on('reconnecting', (error) => {
    if (error.code === 53801) {
      console.log(`Reconnecting your signaling connection!`, error.message);
    } else if (error.code === 53405) {
      console.log(`Reconnecting your media connection!`, error.message);
    }
    updateRoomState(room.state);
  });
}

exports.setupReconnectionUpdates = setupReconnectionUpdates
```

Room state is:

Connected

Disconnected

Reconnecting

Create Room

Leave Room

Connected

Reconnecting

Reconnecting your media connection! Media connection failed or index.js:31
Media activity ceased index.js:23
Reconnected to the Room! index.js:23
Reconnecting your signaling connection! Signaling connection disconnected index.js:29
Reconnected to the Room! index.js:23

Room Sid: RM3107ab7b16929528588232278a2b2480

Enabling and Disabling Tracks

The screenshot shows a browser window at `localhost:3000/localmediacontrols/`. On the left, there is a code editor with JavaScript code for managing media tracks. On the right, there is a developer tools console tab showing some log output.

```
use strict';

/**
 * Mute/unmute your media in a Room.
 * @param {Room} room - The Room you have joined
 * @param {'audio'|'video'} kind - The type of media you want to mute/unmute
 * @param {'mute'|'unmute'} action - Whether you want to mute/unmute
 */
function muteOrUnmuteYourMedia(room, kind, action) {
  const publications = kind === 'audio'
    ? room.localParticipant.audioTracks
    : room.localParticipant.videoTracks;

  publications.forEach(function(publication) {
    if (action === 'mute') {
      publication.track.disable();
    } else {
      publication.track.enable();
    }
  });
}

/**
 * Mute your audio in a Room.
 * @param {Room} room - The Room you have joined
 * @returns {void}
 */
function muteYourAudio(room) {
  muteOrUnmuteYourMedia(room, 'audio', 'mute');
}

/**
 * Mute your video in a Room.
 * @param {Room} room - The Room you have joined
 * @returns {void}
 */
function muteYourVideo(room) {
  muteOrUnmuteYourMedia(room, 'video', 'mute');
}

/**
 * Unmute your audio in a Room.
 * @param {Room} room - The Room you have joined
 * @returns {void}
 */
function unmuteYourAudio(room) {
  muteOrUnmuteYourMedia(room, 'audio', 'unmute');
}
```

Local Media Controls

Enable Audio Enable Video

RemoteParticipant View

Console tab in developer tools:

```
11 Issues: 10 1
```

Room Sid: RM2cbacc90d60fd14ece01c7fa365bd1b6

Data Tracks

The screenshot shows a browser window at `localhost:3000/datatracks/`. It displays a messaging interface between two participants, P1 and P2, using LocalDataTracks.

P1 (left):

```
'use strict';

var Video = require('twilio-video');

/**
 * Connect to the given Room with a LocalDataTrack.
 * @param {string} token - AccessToken for joining the Room
 * @returns {CancelablePromise<Room>}
 */
async function connectToRoomWithDataTrack(token, roomName) {
  const localDataTrack = new Video.LocalDataTrack({
    name: 'chat',
  });

  const room = await Video.connect(token, {
    name: roomName,
    tracks: [localDataTrack],
  });

  return room;
}

/**
 * Send a chat message using the given LocalDataTrack.
 * @param {LocalDataTrack} dataTrack - The <a href="LocalDataTrack"> to send
 * @param {string} message - The message to be sent
 */
function sendChatMessage(dataTrack, message) {
  dataTrack.send(message);
}

/**
 * Receive chat messages from RemoteParticipants in the given Room.
 * @param {Room} room - The Room you are currently in
 * @param {Function} onMessageReceived - Updates UI when a message is received

```

P2 (right):

P2: has connected
P2: 23434
P1: hi
P2: are you there?

Message

Submit

Message

Submit

Console tab in developer tools:

```
1 issue: 1
```

Room Sid: RMabfc7d39f1715245e7e89bd08cda4386

Video Track Automatic Controls

localhost:3000/autorender.html/

Video Track Automatic Controls

```
use strict;

var Video = require('twilio-video');

/**
 * Connect to a Room with 'auto' mode. This is the default mode.
 * @param {string} token - AccessToken for joining the Room
 * @returns {Room}
 */
function joinRoom(token) {
  return Video.connect(token, {
    name: 'my-cool-room',
    bandwidthProfile: {
      video: {
        contentPreferencesMode: 'auto',
        clientTrackSwitchOffControl: 'auto'
      }
    }
  });
}

module.exports.joinRoom = joinRoom;
```

Remote Video Controls

Toggle Visibility: [Show](#) [Hide](#)

Video Size: [640x480](#)

Remote Video Track

Off

twilio

Video Bitrate

17:23:00 17:24:00 17:25:00 17:26:00 17:27:00

600
400
200
0

Console Sources Network

11 Issues: 10 1

Room Sid: RMd9600cff5ed6debfa8b8edfb2d66dad4