Overview

BlueJeans enables developers to receive individual video streams from each video participant in a meeting. This provides an enhanced remote video quality experience with the resolution and frame rate better when compared to a single composited stream in an earlier hybrid model.

The purpose of the individual video stream control is to stream the video of individual participants of your choice. BlueJeans SDK allows you to build custom layouts to have customized video experiences. By building the custom layouts,

- you can control the individual video streams
- you can choose the resolution for a participant.
- you can prioritize by the active speaker
- you can build support for pinning video tiles of particular participants in the meeting.

The API provides the ability to bind the participant's video stream to a view, with control over the resolution and fps of the video, as well as getting metadata about the participant for UI elements.

Pre-requisites

- Xcode 12.5
- Starter App
- Your meeting must support a custom layout (Ref. Steps for building custom layout).
- You can have a maximum of 25 video streams in the custom layout.

Note

- The requested video stream resolutions can be served on a best-effort basis. BlueJeans will dynamically adapt the resolution to a lower/higher value, as suitable, given your network and hardware capabilities.
- Setting the layout to custom is possible before or during a meeting. Gallery, People, or Speaker layouts can only be set during a meeting.

Supported Platforms

iOS 13 and above

Steps for building custom layout

Do the following steps for building a custom video layout.

- Custom layout indicates that you are going to override the BlueJeans layout and handle the video requests and layouts. Do the following setting to set the video layout to `.custom`.

```
meetingService.setVideoLayout(to: .custom)
```

- Some meetings do not support custom video layouts, you should observe `MeetingServiceProtocol/videoLayout` if the value is not `.custom`, you will either fallback using the `VideoDeviceServiceProtocol/getRemoteVideoController` or give an appropriate error.

Note

It is not possible to use custom layouts in meetings that do not enable the `iOS_client_layout_ordering` feature string. If this is not enabled for your enterprise, you can contact support team for the `ios_client_layout_ordering` feature to be enabled.

- To receive 720p video in the custom layout, use the following command.

```
videoDeviceService.set720pVideoReceiveEnabled(true)
```

Note

If you want to receive 720p video, you must enable it as `set720pVideoReceiveEnabled` as true. Otherwise, the videos will be limited to 360p.

- Invoke `setConfiguration()` API, to specify the video stream resolution quality that you desire (for example, five videos with 360p resolution and two videos with 720p resolution).
- Observe the video stream array that you set in the above step. Observing the video arrays is important, because if any change in the `setconfiguration()` that request 720p stream for one of the participant and then the video stream is null or 180p a couple seconds jump up to 720p here you will get three callbacks based on the configurations.
- Based on the video stream, you can create your custom UI (See. VideoStream and Starter App).
    - Create container views for your videos to go in the layout on the screen (See. Rendering Video).
- Invoke the attach and detach methods to render a participant video view.

```
videoStreamService.attachParticipantStreamToView(participantId: newStream.participantId,
view: self. videoContainer
```

Video Stream Service

The `VideoStreamService` provides an API related to requesting a custom configuration of streams of remote video. You can use `VideoStreamService` to create custom video layouts and enable use cases which the included `getRemoteVideoController()` is not suitable.

setVideoStreamConfiguration(configuration:)

Using `setVideoStreamConfiguration`, you can request the video configuration received by the client. You can control the following with `setVideoStreamConfiguration`.

- You can specify the number of video streams to be received.
- You can specify the qualities of the video streams.
- You can specify the video stream priority
- You can specify which participants to include in the stream.

```
func setVideoStreamConfiguration(configuration: [VideoStreamConfiguration]) ->
VideoStreamConfigurationResult
```

Note

BlueJeans cannot guarantee the requested resolution. This will be done on a best effort basis.

Attach Participant Stream

Use `attachParticipantStreamToView` method to attach a `VideoStreamProtocol` to your UIView. This method renders the video stream and displays it in the specified view.

```
func attachParticipantStreamToView(participantId: String, view: UIView) ->
AttachParticipantStreamResult
```

Note

It should only be called for videos in the `videoStreams` array.

Rendering Video

The scaling of the rendered video is defined using `UIView.ContentMode` property of the `VideoView`, you may choose between `.scaleAspectFit` , `.scaleAspectFill` or `.scaleToFill` .

The content mode for attaching video participants. The `UIView.ContentMode` property can be set on views individually after attaching. This will change how the video is rendered within the provided UIView. Setting the mode will only affect views attached after the default is set. The supported values are `.scaleToFill`, `.scaleAspectFit`, `.scaleAspectFill`.

`setDefaultVideoContentMode(_ contentMode: UIView.ContentMode)` can be used to choose a layout where videos maintain their aspect ratio but are cropped within their container, by setting the content mode to `.scaleAspectFill`. Alternatively, you can maintain the video aspect ratio with a clear space on the sides, or top and bottom by setting the content mode to `.scaleAspectFit`.

By default, it is `.scaleAspectFit` (which means “pillar boxing”, or “letterboxing”). BlueJeans SDK allows you to render a video of your choice. If you want the entire video to fill the screen, you can switch to `.scaleAspectFill`. Use the following configuration to switch the `.contentMode` between `.scaleAspectFit` and `.scaleAspectFill` on double tap (when the videoDoubleTapped @IBAction is triggered).

```
 @IBAction func videoDoubleTapped(_ sender: Any) {
       if let currentVideoView = currentVideoView, currentVideoView.contentMode == .scaleAspectFit {
              currentVideoView.contentMode = .scaleAspectFit
           } else {
               currentVideoView?.contentMode = .scaleAspectFill
           }
   }
```

You should consider your layout and decide whether cropping or letterboxing is more appropriate for your use case. You can also observe the resolution of the stream and resize the container instead.

Video Stream

The `videoStreams` array will only be populated with streams of video from participants with video on. For the best meeting experience, it is important to consider both participants with video only and those with audio.

You may use the observable array of participants in the `ParticipantsService` to observe calculate variables such as join time, mute states, and time spent as dominant speaker. Using these variables to inform the ordering of participants in the meeting.

A sample scenario - a remote learning use case

While the pre-built layout takes away all the resize complexity from the developer. Sometimes the scenarios require specific layout configurations. For example, you may have a virtual classroom where you would like the teachers' video to take a bigger share of the view space compared to other student video tiles.

In such cases, the pre-built layout options may not be suitable for your application. In which case you can choose to layout the individual video tiles yourself. The BlueJeans SDK will still take care of rendering the video, you can still choose which remote participant video to render, at what size and where it will lie in your application UI (See. Remote Education and AdvancedCustomLayoutViewController.swift - name of the file in the HelloBlueJeans app).

Ordering use case specific

Using a custom layout, you can make ordering the participants. For example, in an education focused setting you may prefer to sort alphabetically. In a presentation you may wish to highlight a specific speaker. This can be achieved by building on top of the information from both the `VideoStreamService` and the `ParticipantsService`.

Primary Speaker
For some scenarios, such as team collaboration use cases, you might want to equally distribute the space for all video tiles. While in other scenarios, such as a primary speaker use case, you may want to optimize the space for one source. The framework allows you to create the right layout for your scenario (Ref. Video Stream Priority enumeration).

Note

BlueJeans SDK Framework allows you to build the right layout for your scenario.

Choosing Configuration

To choose a configuration for video streams, you should carefully consider the below use cases, intended layout and how the app will be used. Different configurations will have different effects on bandwidth consumption, performance, and battery life.

One-to-one call (TeleHealth use case)

Telehealth is the use of digital information and communication technologies to remotely access medical services and manage medical care. Depending on your use case, you can customize the layout for telehealth sessions, video appointments, virtual consultations, group therapy, and more to your choice. A sample use case (1-1) has been listed below.
In your use case, you just have one remote video is necessary where video is filled the entire screen. In this scenario, the video is large and may require a high quality stream.

```
let configuration = [VideoStreamConfiguration(
   requestedStreamQuality: .r720p_30fps,
   streamPriority: .high
)]
```

In the above use case, `participantID` is not required.

You can alter the stream quality with the high, medium, and low quality of your choice.

Note
BlueJeans SDK downgrades the video stream when it exceeds the available bandwidth. We recommend that you request only the quality that you need. Otherwise, the user's bandwidth will be used more than necessary.

Remote Education

In your use case, there may be couple of trainers, for simplicity the below use case taken as a one Teacher/moderator using participantID. Use participantID as moderator from the `ParticipantsServiceProtocol`.

```
func getModeratorId() -> String? {
    let participantsService = BlueJeansSDK.meetingService.participantsService
    let moderator = participantsService.participants.value?.first(where: { participant in
       participant.isModerator == true
    })
    return moderator?.id
}
```

This function assumes there is only one moderator (there may be many) but we will just take the first for simplicity.

```
func getModeratorConfiguration() -> VideoStreamConfiguration? {
  guard let moderatorId = getModeratorId() else {
    print("No moderator?!?")
    return nil
  }
  return VideoStreamConfiguration(requestedStreamQuality: .r360p_30fps, participantId: moderatorId,
                                  streamPriority: .high)
 }
```

For the other participants/students, you can request a grid of small tiles. In this use case, the moderator/ teacher will be displayed with the highest priority, and the rest participants/students will be displayed with lower resolution.

```
func getStudentsConfiguration(numberOfVideos: Int = 24) -> [VideoStreamConfiguration] {
  let oneVideo = VideoStreamConfiguration(requestedStreamQuality: .r90p_7fps, streamPriority: .medium)
  let allVideos = [VideoStreamConfiguration](repeating: oneVideo, count: numberOfVideos)
  return allVideos
}
```

To get the total configuration of videos that requested by combining the two for one total configuration.

```
   func getTotalConfiguration() -> [VideoStreamConfiguration]? {
       guard let moderatorConfiguration = getModeratorConfiguration() else {
           print("No moderator :(")
           return nil
       }
       return [moderatorConfiguration] + getStudentsConfiguration()
   }
```

Video Stream Priority enumeration

Stream prioritization provides you to have more control over the playback video by allowing you to specify priority order for the streams. The video streams are prioritized is as follows:

- The high-priority video stream receives the video of the most recent speaking participant.
- Higher priority streams are allocated bandwidth before lower priority streams.
- The overall configuration is ordered from highest to lowest priority.

The stream priority is classified as follows

- High (Participants with higher priority will be displayed in high resolution)
- Medium (Participants with medium priority will be displayed in medium resolution)
- Low (Participants with low priority will be displayed in lower resolution)

In order to guarantee a particular stream will receive a higher resolution than other streams, the stream should have a distinct, and higher priority than the other streams.

The below use case explains the scenario where stream priority will be chosen according to the active speakers. An active speaker will be shown on the screen with 720p, the preceding speaker will be shown 360p with 30fps, the earlier preceding speaker will be shown 360p with 15 fps, and 6 more participants shown 180p 15 fps with low stream priority.

```
    let configuration = [
          VideoStreamConfiguration(requestedStreamQuality: .r720p_30fps, streamPriority: .high),
          VideoStreamConfiguration(requestedStreamQuality: .r360p_30fps, streamPriority: .medium),
          VideoStreamConfiguration(requestedStreamQuality: .r360p_15fps, streamPriority: .medium),
          VideoStreamConfiguration(requestedStreamQuality: .r180p_15fps, streamPriority: .low),
          VideoStreamConfiguration(requestedStreamQuality: .r180p_15fps, streamPriority: .low),
          VideoStreamConfiguration(requestedStreamQuality: .r180p_15fps, streamPriority: .low),
          VideoStreamConfiguration(requestedStreamQuality: .r180p_15fps, streamPriority: .low),
          VideoStreamConfiguration(requestedStreamQuality: .r180p_15fps, streamPriority: .low),
          VideoStreamConfiguration(requestedStreamQuality: .r180p_15fps, streamPriority: .low)
    ]
```

Note

`invalidStreamPriorityCombination` is a `videoStreamConfiguration` error that occurs when you tried to mix a configuration with "high, medium, low" priority and a configuration with a priority of "none" (e.g., `streamPriority` is not given or set to nil). The configuration must specify all priorities or no priorities.

User Experience Recommendation

- You can only have one video view per stream of video, i.e., you cannot render the same video in two different views on screen.
- You can request a specific video quality, but it may not be provided depending on the network or other factors.

Note

You should only request the quality you require (e.g. The 720p video should not be requested for a small thumbnail because it is not necessary, consumes more bandwidth, and consumes more power).

Known Limitations

The "Video share and upload feature" found on some BlueJeans Clients is not supported when using custom layouts.