



Panduan Pengguna Streaming Waktu Nyata

Amazon IVS



Amazon IVS: Panduan Pengguna Streaming Waktu Nyata

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Merek dagang dan tampilan dagang Amazon tidak boleh digunakan sehubungan dengan produk atau layanan apa pun yang bukan milik Amazon, dengan cara apa pun yang dapat menyebabkan kebingungan di antara para pelanggan, atau dengan cara apa pun yang menghina atau mendiskreditkan Amazon. Semua merek dagang lain yang tidak dimiliki oleh Amazon merupakan properti dari masing-masing pemilik, yang mungkin berafiliasi, terkait dengan, atau disponsori oleh Amazon, atau tidak.

Table of Contents

Apa itu Streaming Real-Time IVS?	1
Solusi Global, Kontrol Regional	2
Streaming dan Melihat Global	2
Kontrol adalah Regional	2
Memulai dengan IVS	4
Pengantar	4
Prasyarat	4
Referensi Lainnya	4
Terminologi Streaming Waktu Nyata	5
Ikhtisar Langkah	5
Mengatur Izin IAM	6
Menggunakan Kebijakan yang Ada untuk Izin IVS	6
Opsional: Buat Kebijakan Kustom untuk Izin Amazon IVS	7
Buat Pengguna Baru dan Tambahkan Izin	8
Menambahkan Izin ke Pengguna yang Ada	9
Buat Panggung	10
Instruksi Konsol	10
Instruksi CLI	11
Mendistribusikan Token Peserta	12
Instruksi Konsol	13
Instruksi CLI	13
AWSInstruksi SDK	14
Integrasikan SDK Siaran IVS	14
Web	15
Android	15
iOS	16
Publikasikan dan Berlangganan Video	18
Web	18
Android	26
iOS	51
Pemantauan	82
Apa itu Sesi Panggung?	82
Lihat Sesi Panggung dan Peserta	82
Instruksi Konsol	82

Lihat Acara untuk Peserta	82
Instruksi Konsol	82
Instruksi CLI	83
CloudWatchMetrik Akses	84
CloudWatch Instruksi Konsol	84
Instruksi CLI	85
CloudWatch Metrik: Streaming Waktu Nyata IVS	85
SDK Siaran IVS	90
Persyaratan Platform	91
Platform Asli	91
Browser Desktop	91
Browser Seluler (iOS dan Android)	92
Tampilan Web	92
Akses Perangkat yang Diperlukan	92
Dukungan	92
Versioning	93
Panduan Web	93
Memulai	94
Menerbitkan dan Berlangganan	97
Masalah dan Solusi yang Diketahui	109
Penanganan Kesalahan	111
Panduan Android	114
Memulai	115
Penerbitan dan Berlangganan	116
Masalah dan Solusi yang Diketahui	126
Penanganan Kesalahan	128
Panduan iOS	131
Memulai	131
Penerbitan dan Berlangganan	134
Bagaimana iOS Memilih Resolusi Kamera dan Frame Rate	142
Masalah dan Solusi yang Diketahui	143
Penanganan Kesalahan	144
Sumber Gambar Kustom	147
Android	148
iOS	148
Filter Kamera Pihak Ketiga	149

Mengintegrasikan Filter Kamera Pihak Ketiga	149
BytePlus	150
DeepAR	152
Jepret	152
Penggantian Latar Belakang	167
Mode Audio Seluler	189
Pengantar	189
Preset Mode Audio	190
Kasus Penggunaan Tingkat Lanjut	193
Integrasi dengan SDK Lain	194
Menggunakan Amazon EventBridge dengan IVS	195
Membuat EventBridge Aturan Amazon untuk Amazon IVS	196
Contoh: Perubahan Status Komposisi	196
Contoh: Pembaruan Tahap	200
Komposisi Sisi Server	202
ManfaatIT	202
IVS API	203
Layout	204
Memulai	205
Prasyarat	205
Instruksi CLI	206
Aktifkan Berbagi Layar	208
Siklus Hidup Komposisi	212
Rekaman Komposit	214
.....	214
Prasyarat	214
Contoh Pererekaman Komposit: StartComposition dengan Tujuan Bucket S3	215
Isi Rekaman	217
Kebijakan Bucket untuk StorageConfiguration	218
File Metadata JSON	219
Contoh: recording-started.json	221
Contoh: recording-ended.json	222
Contoh: perekaman-failed.json	222
Pemutaran Konten Rekaman dari Bucket Pribadi	223
Menyiapkan Pemutaran menggunakan CloudFront CORS Diaktifkan	224
Contoh: Kebijakan Bucket S3 dengan CloudFront dan Akses IVS	227

Memecahkan masalah	228
Masalah yang Diketahui	228
OBS dan WHIP Support	229
Panduan OBS	229
Kuota Layanan	231
Kuota Layanan Meningkat	231
Kuota Tingkat Panggilan API	231
.....	231
Kuota Lainnya	232
.....	232
Pengoptimalan Streaming	235
Pengantar	235
Streaming Adaptif: Pengkodean Berlapis dengan Simulcast	235
Lapisan Default, Kualitas, dan Framerate	236
Mengkonfigurasi Layered Encoding dengan Simulcast	237
Konfigurasi Streaming	238
Mengubah Bitrate Stream Video	238
Mengubah Framerate Stream Video	239
Mengoptimalkan Audio Bitrate dan Dukungan Stereo	240
Optimasi yang Disarankan	241
Sumber Daya & Dukungan	242
Sumber daya	242
Demo	242
Dukungan	243
Glosarium	244
Riwayat Dokumen	264
Perubahan Panduan Pengguna Streaming Waktu Nyata	264
Perubahan Referensi API Streaming Waktu Nyata IVS	276
Catatan rilis	278
Februari 6, 2024	278
OBS dan WHIP Support	278
Februari 1, 2024	278
SDK Siaran Amazon IVS: Android 1.14.1, iOS 1.14.1, Web 1.8.0 (Streaming Waktu Nyata)	278
Januari 3, 2024	281

SDK Siaran Amazon IVS: Android 1.13.4, iOS 1.13.4, Web 1.7.0 (Streaming Waktu Nyata)	281
Desember 7, 2023	283
CloudWatch Metrik Baru	283
Desember 4, 2023	283
SDK Siaran Amazon IVS: Android 1.13.2 dan iOS 1.13.2 (Streaming Waktu Nyata)	283
21 November 2023	284
SDK Siaran Amazon IVS: Android 1.13.1 (Streaming Waktu Nyata)	284
17 November 2023	285
SDK Siaran Amazon IVS: Android 1.13.0 dan iOS 1.13.0 (Streaming Waktu Nyata)	285
16 November 2023	290
Rekaman Komposit	290
16 November 2023	291
Komposisi Sisi Server	291
16 Oktober 2023	292
SDK Siaran Amazon IVS: Web 1.6.0 (Streaming Waktu Nyata)	292
12 Oktober 2023	292
CloudWatch Metrik Baru dan Data Peserta	292
12 Oktober 2023	292
SDK Siaran Amazon IVS: Android 1.12.1 (Streaming Waktu Nyata)	292
14 September 2023	293
SDK Siaran Amazon IVS: Web 1.5.2 (Streaming Waktu Nyata)	293
23 Agustus 2023	294
SDK Siaran Amazon IVS: Web 1.5.1, Android 1.12.0, dan iOS 1.12.0 (Streaming Waktu Nyata)	294
Agustus 7, 2023	296
SDK Siaran Amazon IVS: Web 1.5.0, Android 1.11.0, dan iOS 1.11.0	296
Agustus 7, 2023	298
Streaming Waktu Nyata	298
.....	ccxcix

Apa itu Streaming Real-Time Amazon IVS?

Amazon Interactive Video Service (IVS) Streaming Real-Time memberi Anda semua yang Anda butuhkan untuk menambahkan audio dan video real-time ke aplikasi Anda.

Kekuatan:

- Latensi waktu nyata - Buat aplikasi untuk kasus penggunaan yang sensitif terhadap latensi, membantu pemirsa Anda tetap terhubung dan terlibat dengan streaming real-time IVS. Mengirimkan live stream dengan latensi yang bisa di bawah 300 milidetik dari host ke penampil.
- Kompetisi tinggi - Buka potensi interaksi skala besar dengan streaming real-time IVS. Mengakomodasi pemirsa hingga 10.000 penonton dan memungkinkan hingga 12 tuan rumah untuk mengambil panggung virtual.
- Dioptimalkan untuk seluler - Streaming real-time IVS dioptimalkan untuk kasus penggunaan seluler, melayani beragam perangkat dan kemampuan jaringan. Dengan mengintegrasikan SDK siaran Amazon IVS untuk Android dan iOS, pengguna Anda dapat berinteraksi sebagai host atau pemirsa, menikmati streaming langsung berkualitas tinggi di perangkat seluler mereka.

Gunakan kasus:

- Tempat tamu — Buat aplikasi yang memungkinkan tuan rumah mempromosikan tamu “di atas panggung”, mengubah pemirsa menjadi tuan rumah untuk interaksi real-time.
- Mode Versus (VS) - Menghasilkan pengalaman denganside-by-sidekompetisi dan biarkan pemirsa menonton host bersaing secara real-time.
- Ruang audio - Undang pendengar untuk bergabung dalam percakapan sebagai tamu dan mendorong keterlibatan yang lebih dalam di ruang audio Anda.
- Lelang video langsung - Ubah lelang menjadi acara video interaktif dan pertahankan kegembiraan dan integritasnya dengan latensi waktu nyata.

Selain dokumentasi produk di sini, lihat <https://ivs.rocks/>, situs khusus untuk menelusuri konten yang dipublikasikan (demo, sampel kode, posting blog), perkiraan biaya, dan pengalaman Amazon IVS melalui demo langsung.

Solusi Global, Kontrol Regional

Streaming dan Melihat Global

Anda dapat menggunakan Amazon IVS untuk streaming ke pemirsa di seluruh dunia:

- Saat Anda melakukan streaming, Amazon IVS secara otomatis menyerap video di lokasi di dekat Anda.
- Pemirsa dapat menonton streaming langsung Anda secara global.

Cara lain untuk mengatakan ini adalah bahwa “bidang data” bersifat global. Pesawat data mengacu pada streaming/ingesting dan melihat.

Kontrol adalah Regional

Sementara pesawat data Amazon IVS bersifat global, “bidang kontrol” bersifat regional. Control plane mengacu pada konsol Amazon IVS, API, dan sumber daya (tahapan).

Cara lain untuk mengatakan ini adalah bahwa Amazon IVS adalah “regionalAWSlayanan.” Artinya, sumber daya Amazon IVS di setiap wilayah tidak bergantung pada sumber daya serupa di wilayah lain. Misalnya, tahap yang Anda buat di satu wilayah tidak tergantung pada tahapan yang Anda buat di wilayah lain.

Ketika Anda menggunakan sumber daya (misalnya, membuat panggung), Anda harus menentukan wilayah di mana ia akan dibuat. Selanjutnya, ketika Anda mengelola sumber daya, Anda harus melakukannya dari wilayah yang sama di mana mereka diciptakan.

Jika Anda menggunakan...	Anda menentukan wilayah dengan...
Konsol Amazon IVS	MenggunakanPilih Wilayahdrop-down di kanan atas bilah navigasi.
API IVS Amazon	Menggunakan endpoint layanan yang sesuai. Lihat Referensi API Streaming Waktu Nyata Amazon IVS . (Jika Anda mengakses API melalui SDK, atur SDKregionparameter. Lihat Alat untuk MembangunAWS .)

Jika Anda menggunakan...	Anda menentukan wilayah dengan...
CLI AWS	Baik: <ul style="list-style-type: none"><li data-bbox="472 352 1487 386">• Menambahkan <code>--region <aws-region></code> untuk perintah CLI Anda.<li data-bbox="472 411 1295 445">• Menempatkan wilayah di lokal AndaAWSfile konfigurasi.

Ingat, terlepas dari wilayah di mana panggung dibuat, Anda dapat melakukan streaming ke Amazon IVS dari mana saja, dan pemirsa dapat menonton dari mana saja.

Memulai dengan IVS Real-Time Streaming

Dokumen ini akan membawa Anda melalui langkah-langkah yang terlibat dalam mengintegrasikan Amazon IVS Real-Time Streaming ke dalam aplikasi Anda.

Topik

- [Pengantar](#)
- [Mengatur Izin IAM](#)
- [Buat Panggung](#)
- [Mendistribusikan Token Peserta](#)
- [Integrasikan SDK Siaran IVS](#)
- [Publikasikan dan Berlangganan Video](#)

Pengantar

Prasyarat

Sebelum Anda menggunakan Real-Time Streaming untuk pertama kalinya, selesaikan tugas-tugas berikut. Untuk petunjuk, lihat [Memulai Streaming Latensi Rendah IVS](#).

- Buat Akun AWS
- Mengatur Root dan Pengguna Administratif

Referensi Lainnya

- [Referensi SDK Siaran Web IVS](#)
- [Referensi SDK Siaran Android IVS](#)
- [Referensi SDK Siaran IVS iOS](#)
- [Referensi API Streaming Waktu Nyata IVS](#)

Terminologi Streaming Waktu Nyata

Istilah	Deskripsi
Tahap	Ruang virtual tempat peserta dapat bertukar video secara real time.
Host	Seorang peserta yang mengirim video lokal ke panggung.
Orang yang melihat	Peserta yang menerima video dari tuan rumah.
Peserta	Pengguna yang terhubung ke panggung sebagai host atau penampil.
Token peserta	Token yang mengautentikasi peserta saat mereka bergabung dengan panggung.
Siaran SDK	Pustaka klien yang memungkinkan peserta mengirim dan menerima video.

Ikhtisar Langkah

1. [the section called “Mengatur Izin IAM”](#)— Buat kebijakan AWS Identity and Access Management (IAM) yang memberi pengguna serangkaian izin dasar dan menetapkan kebijakan tersebut kepada pengguna.
2. [Buat panggung](#) — Buat ruang virtual tempat peserta dapat bertukar video secara real time.
3. [Mendistribusikan token peserta](#) — Kirim token ke peserta sehingga mereka dapat bergabung dengan panggung Anda.

4. [Integrasikan SDK Siaran IVS](#) — Tambahkan SDK siaran ke aplikasi Anda untuk memungkinkan peserta mengirim dan menerima video: [the section called “Web”](#), [the section called “Android”](#) dan [the section called “iOS”](#)
5. [Publikasikan dan berlangganan video](#) — Kirim video Anda ke panggung dan terima video dari host lain: [the section called “Web”](#), [the section called “Android”](#), dan [the section called “iOS”](#).

Mengatur Izin IAM

Selanjutnya, Anda harus membuat kebijakan AWS Identity and Access Management (IAM) yang memberi pengguna serangkaian izin dasar (misalnya, untuk membuat tahap Amazon IVS dan membuat token peserta) dan menetapkan kebijakan tersebut kepada pengguna. [Anda dapat menetapkan izin saat membuat pengguna baru atau menambahkan izin ke pengguna yang sudah ada](#). Kedua prosedur diberikan di bawah ini.

Untuk informasi selengkapnya (misalnya, untuk mempelajari tentang pengguna dan kebijakan IAM, cara melampirkan kebijakan ke pengguna, dan cara membatasi apa yang dapat dilakukan pengguna dengan Amazon IVS), lihat:

- [Membuat Pengguna IAM di Panduan Pengguna IAM](#)
- Informasi di [Amazon IVS Security](#) tentang IAM dan “Kebijakan Terkelola untuk IVS.”
- Informasi IAM di [Amazon IVS Security](#)

Anda dapat menggunakan kebijakan terkelola AWS yang ada untuk Amazon IVS atau membuat kebijakan baru yang menyesuaikan izin yang ingin Anda berikan kepada sekumpulan pengguna, grup, atau peran. Kedua pendekatan tersebut dijelaskan di bawah ini.

Menggunakan Kebijakan yang Ada untuk Izin IVS

Dalam kebanyakan kasus, Anda akan ingin menggunakan kebijakan terkelola AWS untuk Amazon IVS. Mereka dijelaskan sepenuhnya di bagian [Kebijakan Terkelola untuk IVS](#) Keamanan IVS.

- Gunakan kebijakan terkelola `IVSReadOnlyAccess` AWS untuk memberi pengembang aplikasi Anda akses ke semua titik akhir API Dapatkan dan Daftar IVS (untuk streaming latensi rendah dan real-time).
- Gunakan kebijakan terkelola `IVSFullAccess` AWS untuk memberi pengembang aplikasi Anda akses ke semua titik akhir API IVS (untuk streaming latensi rendah dan real-time).

Opsional: Buat Kebijakan Kustom untuk Izin Amazon IVS

Ikuti langkah-langkah ini:

1. Masuk ke Konsol Manajemen AWS dan buka konsol IAM di <https://console.aws.amazon.com/iam/>.
2. Pada panel navigasi, pilih Kebijakan, lalu pilih Buat kebijakan. Jendela Tentukan izin terbuka..
3. Di jendela Tentukan izin, pilih tab JSON, lalu salin dan tempel kebijakan IVS berikut ke area teks editor kebijakan. (Kebijakan ini tidak mencakup semua tindakan Amazon IVS. Anda dapat menambahkan/menghapus (Allow/Deny) izin akses titik akhir sesuai kebutuhan. Lihat [Referensi API Streaming Waktu Nyata IVS](#) untuk detail tentang titik akhir IVS.)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ivs:CreateStage",
        "ivs:CreateParticipantToken",
        "ivs:GetStage",
        "ivs:GetStageSession",
        "ivs:ListStages",
        "ivs:ListStageSessions",
        "ivs:CreateEncoderConfiguration",
        "ivs:GetEncoderConfiguration",
        "ivs:ListEncoderConfigurations",
        "ivs:GetComposition",
        "ivs:ListCompositions",
        "ivs:StartComposition",
        "ivs:StopComposition"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "cloudwatch:DescribeAlarms",
        "cloudwatch:GetMetricData",
        "s3:DeleteBucketPolicy",
        "s3:GetBucketLocation",
        "s3:GetBucketPolicy",
        "s3:PutBucketPolicy",
```

```

        "servicequotas:ListAWSDefaultServiceQuotas",
        "servicequotas:ListRequestedServiceQuotaChangeHistoryByQuota",
        "servicequotas:ListServiceQuotas",
        "servicequotas:ListServices",
        "servicequotas:ListTagsForResource"
    ],
    "Resource": "*"
}
]
}

```

4. Masih di jendela Tentukan izin, pilih Berikutnya (gulir ke bagian bawah jendela untuk melihat ini). Jendela Review dan Create terbuka.
5. Pada jendela Tinjau dan buat, masukkan nama Kebijakan dan tambahkan Deskripsi secara opsional. Catat nama kebijakan, karena Anda akan membutuhkannya saat membuat pengguna (di bawah). Pilih Buat kebijakan (di bagian bawah jendela).
6. Anda dikembalikan ke jendela konsol IAM, di mana Anda akan melihat spanduk yang mengonfirmasi bahwa kebijakan baru Anda telah dibuat.

Buat Pengguna Baru dan Tambahkan Izin

Kunci Akses Pengguna IAM

Kunci akses IAM terdiri dari ID kunci akses dan kunci akses rahasia. Mereka digunakan untuk menandatangani permintaan terprogram yang Anda buat ke AWS. Jika Anda tidak memiliki kunci akses, Anda dapat membuatnya dari AWS Management Console. Sebagai praktik terbaik, jangan membuat kunci akses root-user.

Satu-satunya waktu Anda dapat melihat atau mengunduh kunci akses rahasia adalah ketika Anda membuat kunci akses. Anda tidak dapat memulihkannya nanti. Namun, Anda dapat membuat kunci akses baru kapan saja; Anda harus memiliki izin untuk melakukan tindakan IAM yang diperlukan.

Selalu simpan kunci akses dengan aman. Jangan pernah membagikannya dengan pihak ketiga (bahkan jika pertanyaan tampaknya datang dari Amazon). Untuk informasi lebih lanjut, lihat [Mengelola access key untuk pengguna IAM](#) dalam Panduan Pengguna IAM.

Prosedur

Ikuti langkah-langkah ini:

1. Di panel navigasi, pilih Pengguna, lalu pilih Buat pengguna. Jendela Tentukan detail pengguna terbuka.
2. Di jendela Tentukan detail pengguna:
 - a. Di bawah Rincian pengguna, ketik nama Pengguna baru yang akan dibuat.
 - b. Periksa Berikan akses pengguna ke AWS Management Console.
 - c. Di bawah Kata sandi konsol, pilih Kata sandi yang dibuat otomatis.
 - d. Periksa Pengguna harus membuat kata sandi baru saat masuk berikutnya.
 - e. Pilih Berikutnya. Jendela Setel izin terbuka.
3. Di bawah Setel izin, pilih Lampirkan kebijakan secara langsung. Jendela kebijakan izin terbuka.
4. Di kotak pencarian, masukkan nama kebijakan IVS (baik kebijakan terkelola AWS atau kebijakan kustom yang Anda buat sebelumnya). Ketika ditemukan, centang kotak untuk memilih kebijakan.
5. Pilih Berikutnya (di bagian bawah jendela). Jendela Review dan Create terbuka.
6. Pada jendela Tinjau dan buat, konfirmasikan bahwa semua detail pengguna sudah benar, lalu pilih Buat pengguna (di bagian bawah jendela).
7. Jendela Ambil kata sandi terbuka, berisi detail login Konsol Anda. Simpan informasi ini dengan aman untuk referensi di masa mendatang. Setelah selesai, pilih Kembali ke daftar pengguna.

Menambahkan Izin ke Pengguna yang Ada

Ikuti langkah-langkah ini:

1. Masuk ke Konsol Manajemen AWS dan buka konsol IAM di <https://console.aws.amazon.com/iam/>.
2. Di panel navigasi, pilih Pengguna, lalu pilih nama pengguna yang ada untuk diperbarui. (Pilih nama dengan mengkliknya; jangan centang kotak pilihan.)
3. Pada halaman Ringkasan, pada tab Izin, pilih Tambahkan izin. Jendela Tambahkan izin terbuka.
4. Pilih Lampirkan kebijakan yang ada secara langsung. Jendela kebijakan izin terbuka.
5. Di kotak pencarian, masukkan nama kebijakan IVS (baik kebijakan terkelola AWS atau kebijakan kustom yang Anda buat sebelumnya). Ketika kebijakan ditemukan, centang kotak untuk memilih kebijakan.
6. Pilih Berikutnya (di bagian bawah jendela). Jendela Review terbuka.
7. Pada jendela Review, pilih Add Permissions (di bagian bawah jendela).
8. Pada halaman Ringkasan, konfirmasikan bahwa kebijakan IVS telah ditambahkan.

Buat Panggung

Panggung adalah ruang virtual di mana peserta dapat bertukar video secara real time. Ini adalah sumber daya dasar dari Real-Time Streaming API. Anda dapat membuat panggung menggunakan konsol atau CreateStage titik akhir.

Kami merekomendasikan bahwa jika memungkinkan, Anda membuat tahap baru untuk setiap sesi logis dan menghapusnya ketika selesai, daripada menjaga di sekitar tahapan lama untuk kemungkinan penggunaan kembali. Jika sumber daya basi (tahap lama, tidak untuk digunakan kembali) tidak dibersihkan, Anda cenderung mencapai batas jumlah maksimum tahapan lebih cepat.

Instruksi Konsol

1. Buka [konsol Amazon IVS](#).

(Anda juga dapat mengakses konsol Amazon IVS melalui [AWS Management Console](#).)

2. Di panel navigasi kiri, pilih Tahapan, lalu pilih Buat tahap. Jendela Create stage muncul.

Amazon IVS > Video > Stages > Create stage

Create stage [Info](#)

A stage allows participants to send and receive video and audio with others in real time. You can broadcast a stage to a channel, allowing viewers to see and hear stage participants without needing to join the stage directly. [Learn more](#) [↗](#)

▶ How Amazon IVS stages work

Setup

Stage name – *optional*

Maximum length: 128 characters. May include numbers, letters, underscores (`_`) and hyphens (`-`).

▶ Tags [Info](#)

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

Cancel

Create stage

3. Secara opsional masukkan nama Panggung. Pilih Buat tahap untuk membuat panggung. Halaman detail panggung muncul, untuk tahap baru.

Instruksi CLI

Untuk menginstal AWS CLI, lihat [Menginstal atau memperbarui AWS CLI versi terbaru](#).

Sekarang Anda dapat menggunakan CLI untuk membuat dan mengelola sumber daya. API panggung berada di bawah namespace `ivs-realtime`. Misalnya, untuk membuat panggung:

```
aws ivs-realtime create-stage --name "test-stage"
```

Tanggapannya adalah:

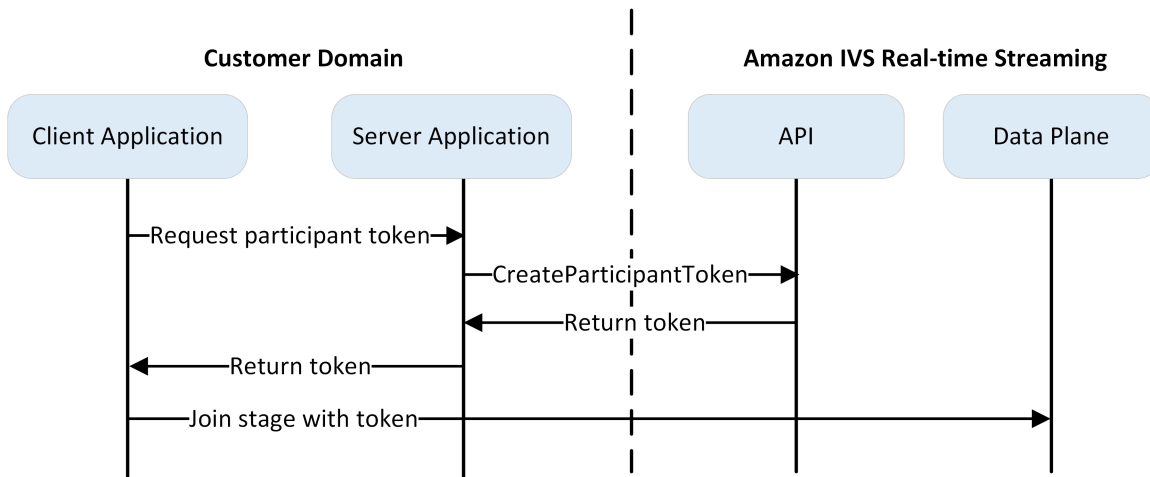
```
{
```

```

"stage": {
  "arn": "arn:aws:ivs:us-west-2:376666121854:stage/VSWjvX5X0kU3",
  "name": "test-stage"
}
}

```

Mendistribusikan Token Peserta



Sekarang setelah Anda memiliki panggung, Anda perlu membuat dan mendistribusikan token kepada peserta untuk memungkinkan mereka bergabung dengan panggung dan mulai mengirim dan menerima video.

Seperti yang ditunjukkan di atas, aplikasi klien meminta aplikasi server Anda untuk token, dan aplikasi server memanggil `CreateParticipantToken` menggunakan permintaan yang ditandatangani AWS SDK atau SigV4. Karena AWS kredensial digunakan untuk memanggil API, token harus dibuat dalam aplikasi sisi server yang aman, bukan aplikasi sisi klien.

Saat membuat token peserta, Anda dapat secara opsional menentukan kemampuan yang diaktifkan oleh token tersebut. Standarnya adalah `PUBLISH` dan `SUBSCRIBE`, yang memungkinkan peserta untuk mengirim dan menerima audio dan video, tetapi Anda dapat mengeluarkan token dengan subset kemampuan. Misalnya, Anda dapat mengeluarkan token hanya dengan `SUBSCRIBE` kemampuan moderator. Dalam hal ini, moderator dapat melihat peserta yang mengirim video tetapi tidak mengirim video mereka sendiri.

Anda dapat membuat token peserta melalui konsol atau CLI untuk pengujian dan pengembangan, tetapi kemungkinan besar Anda ingin membuatnya dengan AWS SDK di lingkungan produksi Anda.

Anda akan memerlukan cara untuk mendistribusikan token dari server Anda ke setiap klien (misalnya, melalui permintaan API). Kami tidak menyediakan fungsi ini. Untuk panduan ini, Anda cukup menyalin dan menempelkan token ke kode klien dalam langkah-langkah berikut.

Penting: Perlakukan token sebagai buram; yaitu, jangan membangun fungsionalitas berdasarkan konten token. Format token bisa berubah di masa depan.

Instruksi Konsol

1. Arahkan ke tahap yang Anda buat pada langkah sebelumnya.
2. Pilih Buat token peserta. Jendela Buat token peserta muncul.
3. Masukkan ID pengguna untuk dikaitkan dengan token. Ini bisa berupa teks yang dikodekan UTF-8.
4. Pilih Buat token peserta.
5. Salin token. **Penting:** Pastikan untuk menyimpan token; IVS tidak menyimpannya dan Anda tidak dapat mengambilnya nanti.

Instruksi CLI

Membuat token dengan AWS CLI mengharuskan Anda mengunduh dan mengonfigurasi CLI terlebih dahulu di mesin Anda. Untuk detailnya, lihat [Panduan Pengguna Antarmuka Baris Perintah AWS](#). Perhatikan bahwa membuat token dengan AWS CLI bagus untuk tujuan pengujian, tetapi untuk penggunaan produksi, kami menyarankan Anda membuat token di sisi server dengan AWS SDK (lihat petunjuk di bawah).

1. Jalankan `create-participant-token` perintah dengan tahap ARN. Sertakan salah satu atau semua kemampuan berikut: "PUBLISH", "SUBSCRIBE".

```
aws ivs-realtime create-participant-token --stage-arn arn:aws:ivs:us-west-2:376666121854:stage/VSWjvX5X0kU3 --capabilities '["PUBLISH", "SUBSCRIBE"]'
```

2. Ini mengembalikan token peserta:

```
{
  "participantToken": {
    "capabilities": [
      "PUBLISH",
      "SUBSCRIBE"
    ],
    "expirationTime": "2023-06-03T07:04:31+00:00",
```


- iOS: <https://github.com/aws-samples/amazon-ivs-real-time-streaming-ios-samples>

Web

Mengatur File

Untuk memulai, atur file Anda dengan membuat folder dan file HTML dan JS awal:

```
mkdir realtime-web-example
cd realtime-web-example
touch index.html
touch app.js
```

Anda dapat menginstal SDK siaran menggunakan tag skrip atau npm. Contoh kami menggunakan tag skrip untuk kesederhanaan tetapi mudah dimodifikasi jika Anda memilih untuk menggunakan npm nanti.

Menggunakan Tag Skrip

SDK siaran Web didistribusikan sebagai JavaScript perpustakaan dan dapat diambil di <https://web-broadcast.live-video.net/1.8.0/amazon-ivs-web-broadcast.js>.

Ketika dimuat melalui <script> tag, perpustakaan mengekspos variabel global dalam lingkup jendela bernama `IVSBroadcastClient`.

Menggunakan npm

Untuk menginstal paket npm:

```
npm install amazon-ivs-web-broadcast
```

Anda sekarang dapat mengakses `BroadcastClient` objek IVS:

```
const { Stage } = IVSBroadcastClient;
```

Android

Buat Proyek Android

1. Di Android Studio, buat Proyek Baru.

2. Pilih Aktivitas Tampilan Kosong.

Catatan: Di beberapa versi Android Studio yang lebih lama, aktivitas berbasis Tampilan disebut Empty Activity. Jika jendela Android Studio menampilkan Aktivitas Kosong dan tidak menampilkan Aktivitas Tampilan Kosong, pilih Aktivitas Kosong. Jika tidak, jangan pilih Aktivitas Kosong, karena kita akan menggunakan View API (bukan Jetpack Compose).

3. Beri nama proyek Anda, lalu pilih Selesai.

Instal Broadcast SDK

Untuk menambahkan library siaran Amazon IVS Android ke lingkungan pengembangan Android Anda, tambahkan pustaka ke `build.gradle` file modul Anda, seperti yang ditunjukkan di sini (untuk versi terbaru SDK siaran Amazon IVS). Dalam proyek yang lebih baru, `mavenCentral` repositori mungkin sudah disertakan dalam `settings.gradle` file Anda, jika itu masalahnya Anda dapat menghilangkan blok `repositories` Untuk sampel kami, kami juga perlu mengaktifkan pengikatan data di android blok.

```
android {
    dataBinding.enabled true
}

repositories {
    mavenCentral()
}

dependencies {
    implementation 'com.amazonaws:ivs-broadcast:1.14.1:stages@aar'
}
```

Sebagai alternatif, untuk menginstal SDK secara manual, unduh versi terbaru dari lokasi ini:

<https://search.maven.org/artifact/com.amazonaws/ivs-broadcast>

iOS

Buat Proyek iOS

1. Buat proyek Xcode baru.
2. Untuk Platform, pilih iOS.

3. Untuk Aplikasi, pilih Aplikasi.
4. Masukkan Nama Produk aplikasi Anda, lalu pilih Berikutnya.
5. Pilih (navigasikan ke) direktori tempat menyimpan proyek, lalu pilih Buat.

Selanjutnya Anda perlu membawa SDK. Kami menyarankan Anda mengintegrasikan SDK siaran melalui CocoaPods. Atau, Anda dapat menambahkan kerangka kerja secara manual ke proyek Anda. Kedua metode dijelaskan di bawah ini.

Direkomendasikan: Instal Broadcast SDK () CocoaPods

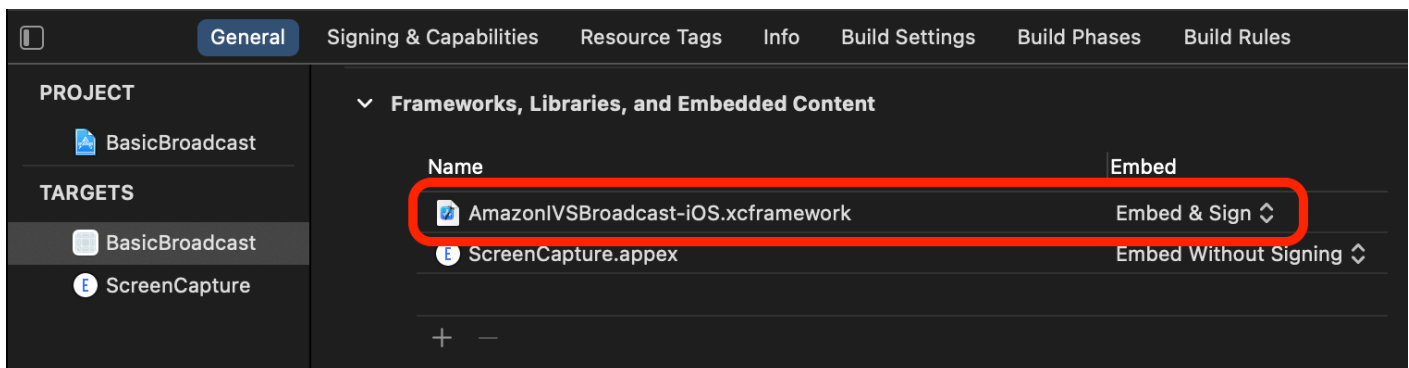
Dengan asumsi nama proyek Anda `BasicRealTime`, buat Podfile di folder proyek dengan konten berikut dan kemudian jalankan `pod install`:

```
target 'BasicRealTime' do
  # Comment the next line if you don't want to use dynamic frameworks
  use_frameworks!

  # Pods for BasicRealTime
  pod 'AmazonIVSBroadcast/Stages'
end
```

Pendekatan Alternatif: Instal Kerangka Secara Manual

1. Unduh versi terbaru dari <https://broadcast.live-video.net/1.14.1/AmazonIVSBroadcast-Stages.xcframework.zip>.
2. Ekstrak isi arsip. `AmazonIVSBroadcast.xcframework` berisi SDK untuk perangkat dan simulator.
3. Sematkan `AmazonIVSBroadcast.xcframework` dengan menyeretnya ke bagian Frameworks, Libraries, dan Embedded Content pada tab General untuk target aplikasi Anda:



Konfigurasi Izin

Anda perlu memperbarui proyek Anda `Info.plist` untuk menambahkan dua entri baru untuk `NSCameraUsageDescription` dan `NSMicrophoneUsageDescription`. Untuk nilainya, berikan penjelasan yang dihadapi pengguna tentang mengapa aplikasi Anda meminta akses kamera dan mikrofon.

Key	Type	Value
Information Property List	Dictionary	(3 items)
Application Scene Manifest	Dictionary	(2 items)
Privacy - Microphone Usage Description	String	We need access to your microphone to publish your audio feed
Privacy - Camera Usage Description	String	We need access to your camera to publish your video feed

Publikasikan dan Berlangganan Video

Lihat detail di bawah ini untuk [web](#), [Android](#), dan [iOS](#).

Web

Buat HTML Boilerplate

Pertama mari kita buat boilerplate HTML dan impor pustaka sebagai tag skrip:

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />

  <!-- Import the SDK -->
  <script src="https://web-broadcast.live-video.net/1.8.0/amazon-ivs-web-
broadcast.js"></script>
</head>

<body>

<!-- TODO - fill in with next sections -->
<script src="./app.js"></script>
```

```
</body>
</html>
```

Terima Masukan Token dan Tambahkan Tombol Gabung/Tinggalkan

Di sini kita mengisi tubuh dengan kontrol input kita. Ini mengambil sebagai masukan token, dan mereka mengatur tombol Gabung dan Tinggalkan. Biasanya aplikasi akan meminta token dari API aplikasi Anda, tetapi untuk contoh ini Anda akan menyalin dan menempelkan token ke input token.

```
<h1>IVS Real-Time Streaming</h1>
<hr />

<label for="token">Token</label>
<input type="text" id="token" name="token" />
<button class="button" id="join-button">Join</button>
<button class="button" id="leave-button" style="display: none;">Leave</button>
<hr />
```

Tambahkan Elemen Kontainer Media

Elemen-elemen ini akan memegang media untuk peserta lokal dan jarak jauh kami. Kita menambahkan tag script untuk memuat logika aplikasi kita didefinisikan dalam `app.js`.

```
<!-- Local Participant -->
<div id="local-media"></div>

<!-- Remote Participants -->
<div id="remote-media"></div>

<!-- Load Script -->
<script src="./app.js"></script>
```

Ini melengkapi halaman HTML dan Anda akan melihat ini saat memuat `index.html` di browser:

IVS Real-Time Streaming

Token

Buat app.js

Mari kita beralih ke mendefinisikan isi `app.js` file kita. Mulailah dengan mengimpor semua properti yang diperlukan dari global SDK:

```
const {
  Stage,
  LocalStageStream,
  SubscribeType,
  StageEvents,
  ConnectionState,
  StreamType
} = IVSBroadcastClient;
```

Buat Variabel Aplikasi

Tetapkan variabel untuk menyimpan referensi ke elemen HTML tombol Gabung dan Tinggalkan kami dan status penyimpanan untuk aplikasi:

```
let joinButton = document.getElementById("join-button");
let leaveButton = document.getElementById("leave-button");

// Stage management
let stage;
let joining = false;
let connected = false;
let localCamera;
let localMic;
let cameraStageStream;
let micStageStream;
```

Buat JoinStage 1: Tentukan Fungsi dan Validasi Input

`joinStageFungsi` mengambil token input, membuat koneksi ke panggung, dan mulai mempublikasikan video dan audio yang diambil dari `getUserMedia`.

Untuk memulai, kita mendefinisikan fungsi dan memvalidasi status dan masukan token. Kami akan menyempurnakan fungsi ini di beberapa bagian berikutnya.

```
const joinStage = async () => {
```

```
if (connected || joining) {
  return;
}
joining = true;

const token = document.getElementById("token").value;

if (!token) {
  window.alert("Please enter a participant token");
  joining = false;
  return;
}

// Fill in with the next sections
};
```

Buat JoinSage 2: Dapatkan Media untuk Publikasikan

Berikut adalah media yang akan dipublikasikan ke panggung:

```
async function getCamera() {
  // Use Max Width and Height
  return navigator.mediaDevices.getUserMedia({
    video: true,
    audio: false
  });
}

async function getMic() {
  return navigator.mediaDevices.getUserMedia({
    video: false,
    audio: true
  });
}

// Retrieve the User Media currently set on the page
localCamera = await getCamera();
localMic = await getMic();

// Create StageStreams for Audio and Video
cameraStageStream = new LocalStageStream(localCamera.getVideoTracks()[0]);
micStageStream = new LocalStageStream(localMic.getAudioTracks()[0]);
```

Buat JoinSage 3: Tentukan Strategi Panggung dan Buat Panggung

Strategi tahap ini adalah inti dari logika keputusan yang digunakan SDK untuk memutuskan apa yang akan dipublikasikan dan peserta mana yang akan berlangganan. Untuk informasi selengkapnya tentang tujuan fungsi, lihat [Strategi](#).

Strategi ini sederhana. Setelah bergabung dengan panggung, publikasikan streaming yang baru saja kami ambil dan berlangganan audio dan video setiap peserta jarak jauh:

```
const strategy = {
  stageStreamsToPublish() {
    return [cameraStageStream, micStageStream];
  },
  shouldPublishParticipant() {
    return true;
  },
  shouldSubscribeToParticipant() {
    return SubscribeType.AUDIO_VIDEO;
  }
};

stage = new Stage(token, strategy);
```

Buat JoinSage 4: Menangani Acara Panggung dan Media Render

Tahapan memancarkan banyak peristiwa. Kita harus mendengarkan dan merender `STAGE_PARTICIPANT_STREAMS_ADDED` dan `STAGE_PARTICIPANT_LEFT` menghapus media ke dan dari halaman. [Serangkaian peristiwa yang lebih lengkap tercantum dalam Acara](#).

Perhatikan bahwa kita membuat empat fungsi pembantu di sini untuk membantu kita dalam mengelola elemen DOM yang diperlukan: `setupParticipant`, `teardownParticipant`, `createVideoEl`, dan `createContainer`.

```
stage.on(StageEvents.STAGE_CONNECTION_STATE_CHANGED, (state) => {
  connected = state === ConnectionState.CONNECTED;

  if (connected) {
    joining = false;
    joinButton.style = "display: none";
    leaveButton.style = "display: inline-block";
  }
});
```

```
stage.on(
  StageEvents.STAGE_PARTICIPANT_STREAMS_ADDED,
  (participant, streams) => {
    console.log("Participant Media Added: ", participant, streams);

    let streamsToDisplay = streams;

    if (participant.isLocal) {
      // Ensure to exclude local audio streams, otherwise echo will occur
      streamsToDisplay = streams.filter(
        (stream) => stream.streamType !== StreamType.VIDEO
      );
    }

    const videoEl = setupParticipant(participant);
    streamsToDisplay.forEach((stream) =>
      videoEl.srcObject.addTrack(stream.mediaStreamTrack)
    );
  }
);

stage.on(StageEvents.STAGE_PARTICIPANT_LEFT, (participant) => {
  console.log("Participant Left: ", participant);
  teardownParticipant(participant);
});

// Helper functions for managing DOM

function setupParticipant({ isLocal, id }) {
  const groupId = isLocal ? "local-media" : "remote-media";
  const groupContainer = document.getElementById(groupId);

  const participantContainerId = isLocal ? "local" : id;
  const participantContainer = createContainer(participantContainerId);
  const videoEl = createVideoEl(participantContainerId);

  participantContainer.appendChild(videoEl);
  groupContainer.appendChild(participantContainer);

  return videoEl;
}
```

```
function teardownParticipant({ isLocal, id }) {
  const groupId = isLocal ? "local-media" : "remote-media";
  const groupContainer = document.getElementById(groupId);
  const participantContainerId = isLocal ? "local" : id;

  const participantDiv = document.getElementById(
    participantContainerId + "-container"
  );
  if (!participantDiv) {
    return;
  }
  groupContainer.removeChild(participantDiv);
}

function createVideoEl(id) {
  const videoEl = document.createElement("video");
  videoEl.id = id;
  videoEl.autoplay = true;
  videoEl.playsInline = true;
  videoEl.srcObject = new MediaStream();
  return videoEl;
}

function createContainer(id) {
  const participantContainer = document.createElement("div");
  participantContainer.classList = "participant-container";
  participantContainer.id = id + "-container";

  return participantContainer;
}
```

Buat JoinSage 5: Bergabunglah dengan Panggung

Mari selesaikan `joinStage` fungsi kita dengan akhirnya bergabung dengan panggung!

```
try {
  await stage.join();
} catch (err) {
  joining = false;
  connected = false;
  console.error(err.message);
}
```

Buat LeaveStage

Tentukan `leaveStage` fungsi yang akan dipanggil tombol tinggalkan.

```
const leaveStage = async () => {
  stage.leave();

  joining = false;
  connected = false;
};
```

Inisialisasi Input-Event Handler

Kami akan menambahkan satu fungsi terakhir ke `app.js` file kami. Fungsi ini dipanggil segera ketika halaman dimuat dan menetapkan event handler untuk bergabung dan meninggalkan panggung.

```
const init = async () => {
  try {
    // Prevents issues on Safari/FF so devices are not blank
    await navigator.mediaDevices.getUserMedia({ video: true, audio: true });
  } catch (e) {
    alert(
      "Problem retrieving media! Enable camera and microphone permissions."
    );
  }

  joinButton.addEventListener("click", () => {
    joinStage();
  });

  leaveButton.addEventListener("click", () => {
    leaveStage();
    joinButton.style = "display: inline-block";
    leaveButton.style = "display: none";
  });
};

init(); // call the function
```


Jalankan Aplikasi dan Berikan Token

Pada titik ini Anda dapat berbagi halaman web secara lokal atau dengan orang lain, [membuka halaman](#), dan memasukkan token peserta dan bergabung dengan panggung.

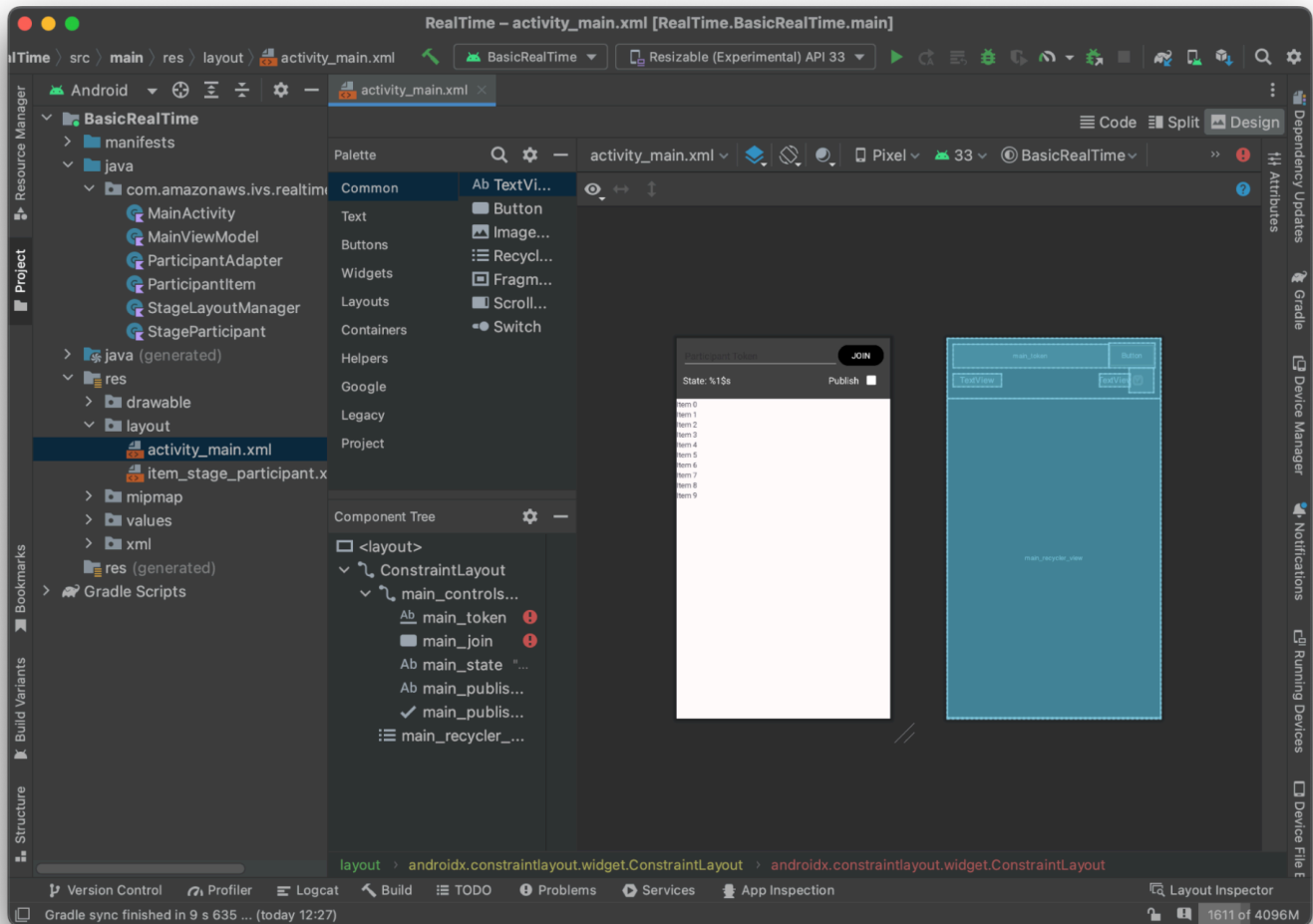
Apa selanjutnya?

Untuk contoh lebih detail yang melibatkan npm, React, dan lainnya, lihat [IVS Broadcast SDK: Web Guide \(Real-Time Streaming Guide\)](#).

Android

Buat Tampilan

Kita mulai dengan membuat tata letak sederhana untuk aplikasi kita menggunakan `activity_main.xml` file yang dibuat secara otomatis. Layout berisi `EditText` to add a token, `JoinButton`, a `TextView` to show the stage state, dan `CheckBox` to toggle publishing.



Berikut adalah XHTML di belakang tampilan:

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:keepScreenOn="true"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".BasicActivity">

        <androidx.constraintlayout.widget.ConstraintLayout
            android:id="@+id/main_controls_container"
            android:layout_width="match_parent"
```

```
android:layout_height="wrap_content"
android:background="@color/cardview_dark_background"
android:padding="12dp"
app:layout_constraintTop_toTopOf="parent">

<EditText
    android:id="@+id/main_token"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:autofillHints="@null"
    android:backgroundTint="@color/white"
    android:hint="@string/token"
    android:imeOptions="actionDone"
    android:inputType="text"
    android:textColor="@color/white"
    app:layout_constraintEnd_toStartOf="@id/main_join"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<Button
    android:id="@+id/main_join"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:backgroundTint="@color/black"
    android:text="@string/join"
    android:textAllCaps="true"
    android:textColor="@color/white"
    android:textSize="16sp"
    app:layout_constraintBottom_toBottomOf="@+id/main_token"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toEndOf="@id/main_token" />

<TextView
    android:id="@+id/main_state"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/state"
    android:textColor="@color/white"
    android:textSize="18sp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@id/main_token" />

<TextView
```

```

        android:id="@+id/main_publish_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/publish"
        android:textColor="@color/white"
        android:textSize="18sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toStartOf="@id/main_publish_checkbox"
        app:layout_constraintTop_toBottomOf="@id/main_token" />

<CheckBox
    android:id="@+id/main_publish_checkbox"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:buttonTint="@color/white"
    android:checked="true"
    app:layout_constraintBottom_toBottomOf="@id/main_publish_text"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintTop_toTopOf="@id/main_publish_text" />

</androidx.constraintlayout.widget.ConstraintLayout>

<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/main_recycler_view"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    app:layout_constraintTop_toBottomOf="@+id/main_controls_container"
    app:layout_constraintBottom_toBottomOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
</layout>

```

Kami mererensikan beberapa ID string di sini, jadi kami akan membuat seluruh `strings.xml` file kami sekarang:

```

<resources>
    <string name="app_name">BasicRealTime</string>
    <string name="join">Join</string>
    <string name="leave">Leave</string>
    <string name="token">Participant Token</string>
    <string name="publish">Publish</string>
    <string name="state">State: %1$s</string>
</resources>

```

Mari kita tautkan pandangan tersebut di XHTML ke: MainActivity.kt

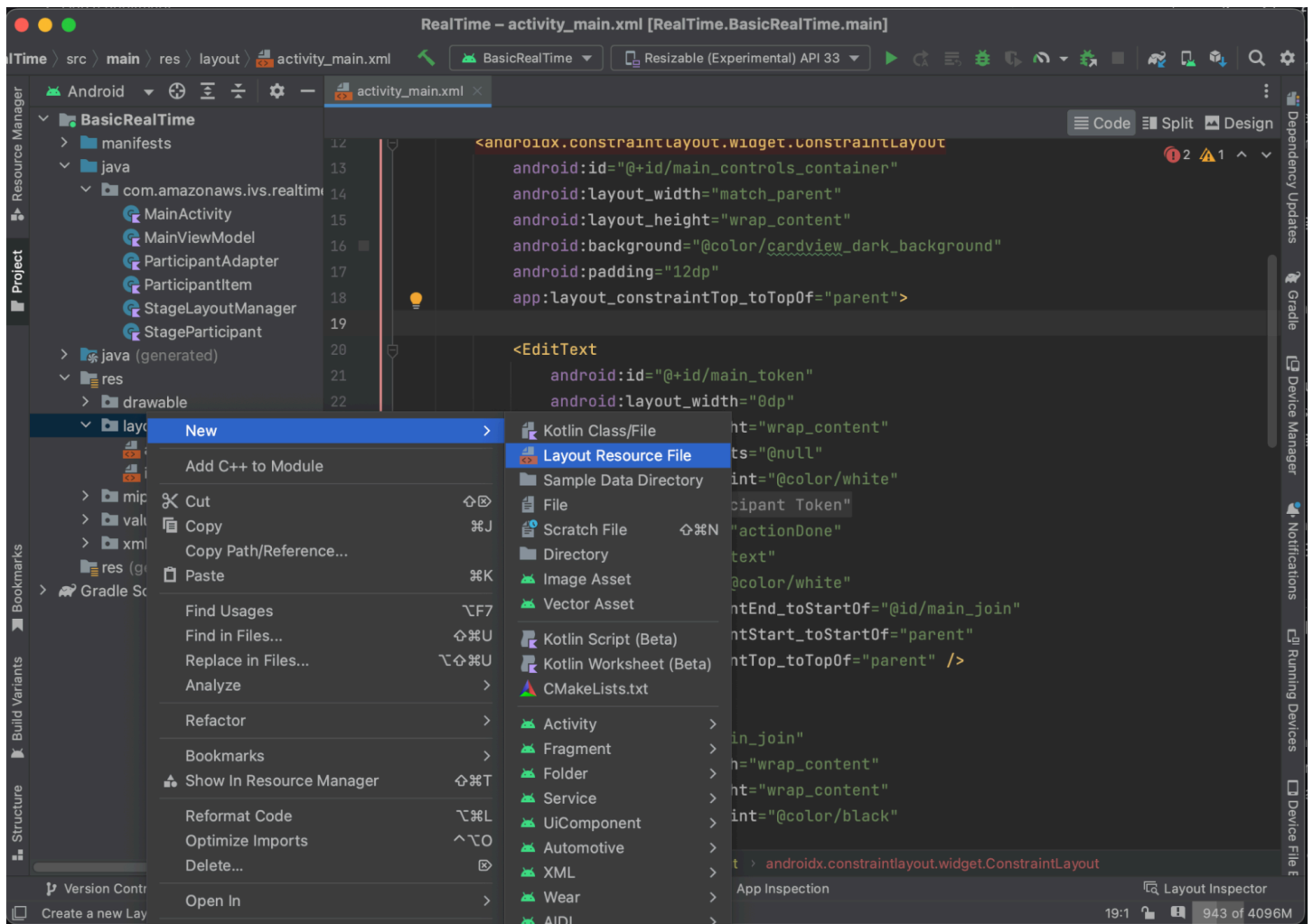
```
import android.widget.Button
import android.widget.CheckBox
import android.widget.EditText
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView

private lateinit var checkBoxPublish: CheckBox
private lateinit var recyclerView: RecyclerView
private lateinit var buttonJoin: Button
private lateinit var textViewState: TextView
private lateinit var editTextToken: EditText

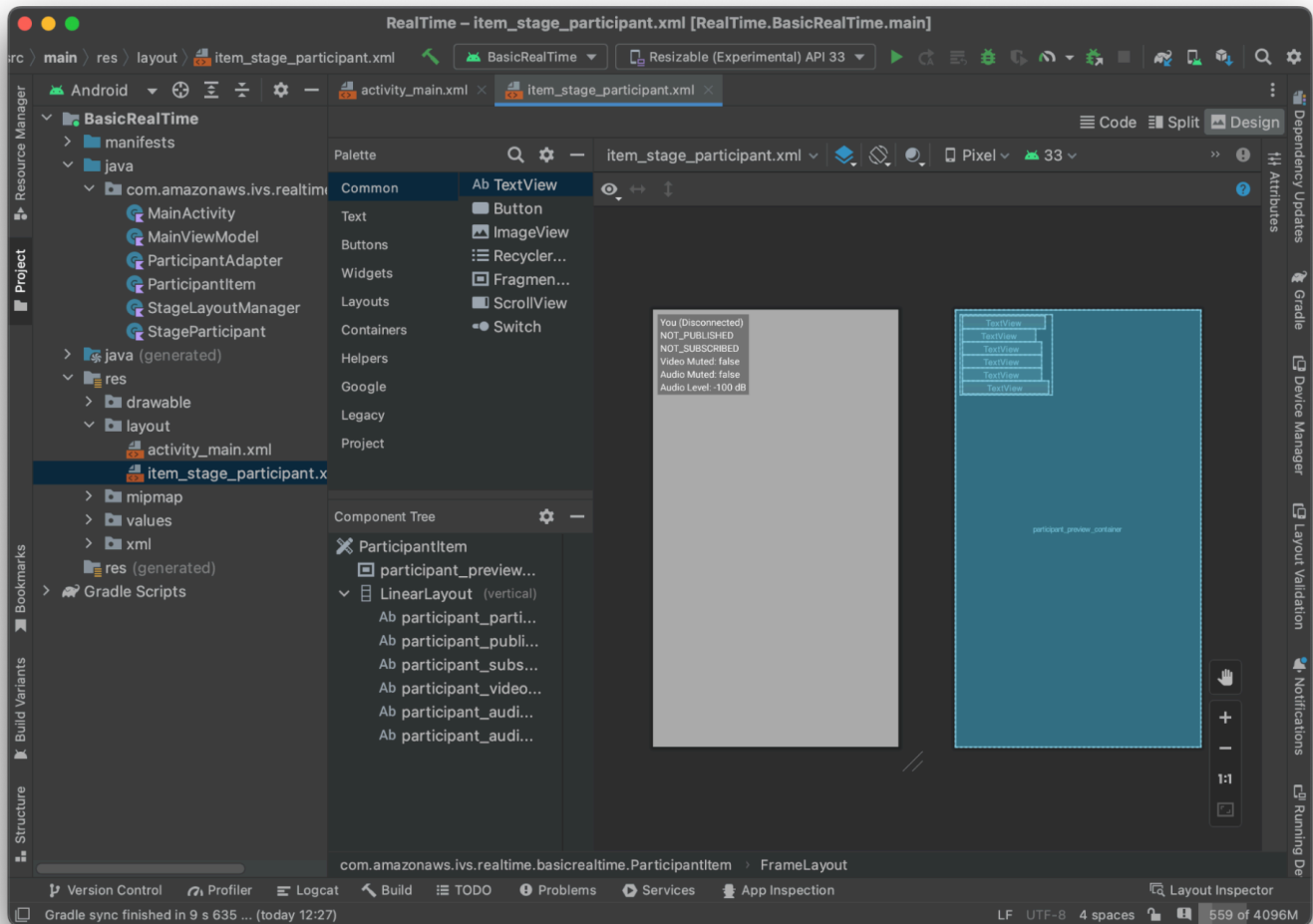
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    checkBoxPublish = findViewById(R.id.main_publish_checkbox)
    recyclerView = findViewById(R.id.main_recycler_view)
    buttonJoin = findViewById(R.id.main_join)
    textViewState = findViewById(R.id.main_state)
    editTextToken = findViewById(R.id.main_token)
}
```

Sekarang kita membuat tampilan item untuk kitaRecyclerView. Untuk melakukan ini, klik kanan res/layout direktori Anda dan pilih New > Layout Resource File. Beri nama file baru iniitem_stage_participant.xml.



Tata letak untuk item ini sederhana: berisi tampilan untuk merender aliran video peserta dan daftar label untuk menampilkan informasi tentang peserta:



Berikut adalah XML-nya:

```
<?xml version="1.0" encoding="utf-8"?>
<com.amazonaws.ivs.realtime.basicrealtime.ParticipantItem xmlns:android="http://
schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <FrameLayout
        android:id="@+id/participant_preview_container"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:background="@android:color/darker_gray" />
```

```
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:background="#50000000"
    android:orientation="vertical"
    android:paddingLeft="4dp"
    android:paddingTop="2dp"
    android:paddingRight="4dp"
    android:paddingBottom="2dp"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">

    <TextView
        android:id="@+id/participant_participant_id"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="@android:color/white"
        android:textSize="16sp"
        tools:text="You (Disconnected)" />

    <TextView
        android:id="@+id/participant_publishing"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="@android:color/white"
        android:textSize="16sp"
        tools:text="NOT_PUBLISHED" />

    <TextView
        android:id="@+id/participant_subscribed"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="@android:color/white"
        android:textSize="16sp"
        tools:text="NOT_SUBSCRIBED" />

    <TextView
        android:id="@+id/participant_video_muted"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="@android:color/white"
        android:textSize="16sp"
```



```
        tools:text="Video Muted: false" />

    <TextView
        android:id="@+id/participant_audio_muted"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="@android:color/white"
        android:textSize="16sp"
        tools:text="Audio Muted: false" />

    <TextView
        android:id="@+id/participant_audio_level"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="@android:color/white"
        android:textSize="16sp"
        tools:text="Audio Level: -100 dB" />

</LinearLayout>

</com.amazonaws.ivs.realtime.basicrealtime.ParticipantItem>
```

File XHTML ini mengembang kelas yang belum kita buat, ParticipantItem Karena XHTML menyertakan namespace lengkap, pastikan untuk memperbarui file XHTML ini ke namespace Anda. Mari kita buat kelas ini dan atur tampilan, tetapi biarkan kosong untuk saat ini.

Buat class Kotlin baru, ParticipantItem:

```
package com.amazonaws.ivs.realtime.basicrealtime

import android.content.Context
import android.util.AttributeSet
import android.widget.FrameLayout
import android.widget.TextView
import kotlin.math.roundToInt

class ParticipantItem @JvmOverloads constructor(
    context: Context,
    attrs: AttributeSet? = null,
    defStyleAttr: Int = 0,
    defStyleRes: Int = 0,
) : FrameLayout(context, attrs, defStyleAttr, defStyleRes) {
```

```
private lateinit var previewContainer: FrameLayout
private lateinit var textViewParticipantId: TextView
private lateinit var textViewPublish: TextView
private lateinit var textViewSubscribe: TextView
private lateinit var textViewVideoMuted: TextView
private lateinit var textViewAudioMuted: TextView
private lateinit var textViewAudioLevel: TextView

override fun onFinishInflate() {
    super.onFinishInflate()
    previewContainer = findViewById(R.id.participant_preview_container)
    textViewParticipantId = findViewById(R.id.participant_participant_id)
    textViewPublish = findViewById(R.id.participant_publishing)
    textViewSubscribe = findViewById(R.id.participant_subscribed)
    textViewVideoMuted = findViewById(R.id.participant_video_muted)
    textViewAudioMuted = findViewById(R.id.participant_audio_muted)
    textViewAudioLevel = findViewById(R.id.participant_audio_level)
}
}
```

Izin

Untuk menggunakan kamera dan mikrofon, Anda perlu meminta izin dari pengguna. Kami mengikuti alur izin standar untuk ini:

```
override fun onStart() {
    super.onStart()
    requestPermission()
}

private val requestPermissionLauncher =
    registerForActivityResult(ActivityResultContracts.RequestMultiplePermissions())
{ permissions ->
    if (permissions[Manifest.permission.CAMERA] == true &&
        permissions[Manifest.permission.RECORD_AUDIO] == true) {
        viewModel.permissionGranted() // we will add this later
    }
}

private val permissions = listOf(
    Manifest.permission.CAMERA,
    Manifest.permission.RECORD_AUDIO,
)
```

```
private fun requestPermission() {
    when {
        this.hasPermissions(permissions) -> viewModel.permissionGranted() // we will
        add this later
        else -> requestPermissionLauncher.launch(permissions.toTypedArray())
    }
}

private fun Context.hasPermissions(permissions: List<String>): Boolean {
    return permissions.all {
        ContextCompat.checkSelfPermission(this, it) ==
        PackageManager.PERMISSION_GRANTED
    }
}
```

Status Aplikasi

Aplikasi kami melacak peserta secara lokal di a `MainViewModel.kt` dan status akan dikomunikasikan kembali ke `MainActivity` menggunakan Kotlin. [StateFlow](#)

Buat class `MainViewModel` Kotlin baru:

```
package com.amazonaws.ivs.realtime.basicrealtime

import android.app.Application
import androidx.lifecycle.AndroidViewModel

class MainViewModel(application: Application) : AndroidViewModel(application),
    Stage.Strategy, Stage.Renderer {

}
```

Dalam `MainActivity.kt` kami mengelola model tampilan kami:

```
import androidx.activity.viewModels

private val viewModel: MainViewModel by viewModels()
```

Untuk menggunakan `AndroidViewModel` dan `ViewModel` ekstensi Kotlin ini, Anda harus menambahkan yang berikut ini ke `build.gradle` file modul Anda:

```
implementation 'androidx.core:core-ktx:1.10.1'
implementation "androidx.activity:activity-ktx:1.7.2"
implementation 'androidx.appcompat:appcompat:1.6.1'
implementation 'com.google.android.material:material:1.10.0'
implementation "androidx.lifecycle:lifecycle-extensions:2.2.0"

def lifecycle_version = "2.6.1"
implementation "androidx.lifecycle:lifecycle-livedata-ktx:$lifecycle_version"
implementation "androidx.lifecycle:lifecycle-viewmodel-ktx:$lifecycle_version"
implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
```

RecyclerView Adaptor

Kami akan membuat `RecyclerView.Adapter` subclass sederhana untuk melacak peserta kami dan memperbarui acara `RecyclerView` di atas panggung kami. Tapi pertama-tama, kita membutuhkan kelas yang mewakili peserta. Buat class `StageParticipant` Kotlin baru:

```
package com.amazonaws.ivs.realtime.basicrealtime

import com.amazonaws.ivs.broadcast.Stage
import com.amazonaws.ivs.broadcast.StageStream

class StageParticipant(val isLocal: Boolean, var participantId: String?) {
    var publishState = Stage.PublishState.NOT_PUBLISHED
    var subscribeState = Stage.SubscribeState.NOT_SUBSCRIBED
    var streams = mutableListOf<StageStream>()

    val stableID: String
        get() {
            return if (isLocal) {
                "LocalUser"
            } else {
                requireNotNull(participantId)
            }
        }
}
```

Kita akan menggunakan kelas ini di `ParticipantAdapter` kelas yang akan kita buat selanjutnya. Kita mulai dengan mendefinisikan kelas dan membuat variabel untuk melacak peserta:

```
package com.amazonaws.ivs.realtime.basicrealtime
```

```
import android.view.LayoutInflater
import android.view.ViewGroup
import androidx.recyclerview.widget.RecyclerView

class ParticipantAdapter : RecyclerView.Adapter<ParticipantAdapter.ViewHolder>() {

    private val participants = mutableListOf<StageParticipant>()
```

Kita juga harus mendefinisikan kita `RecyclerView.ViewHolder` sebelum menerapkan sisa penggantian:

```
class ViewHolder(val participantItem: ParticipantItem) :
    RecyclerView.ViewHolder(participantItem)
```

Dengan menggunakan ini, kita dapat menerapkan `RecyclerView.Adapter` penggantian standar:

```
override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {
    val item = LayoutInflater.from(parent.context)
        .inflate(R.layout.item_stage_participant, parent, false) as ParticipantItem
    return ViewHolder(item)
}

override fun getItemCount(): Int {
    return participants.size
}

override fun getItemId(position: Int): Long =
    participants[position]
        .stableID
        .hashCode()
        .toLong()

override fun onBindViewHolder(holder: ViewHolder, position: Int) {
    return holder.participantItem.bind(participants[position])
}

override fun onBindViewHolder(holder: ViewHolder, position: Int, payloads:
    MutableList<Any>) {
    val updates = payloads.filterIsInstance<StageParticipant>()
    if (updates.isNotEmpty()) {
        updates.forEach { holder.participantItem.bind(it) // implemented later }
    } else {
        super.onBindViewHolder(holder, position, payloads)
    }
}
```

```

    }
}

```

Akhirnya, kami menambahkan metode baru yang akan kami panggil dari kami `MainViewModel` ketika perubahan pada peserta dilakukan. Metode ini adalah operasi CRUD standar pada adaptor.

```

fun participantJoined(participant: StageParticipant) {
    participants.add(participant)
    notifyItemInserted(participants.size - 1)
}

fun participantLeft(participantId: String) {
    val index = participants.indexOfFirst { it.participantId == participantId }
    if (index != -1) {
        participants.removeAt(index)
        notifyItemRemoved(index)
    }
}

fun participantUpdated(participantId: String?, update: (participant: StageParticipant)
-> Unit) {
    val index = participants.indexOfFirst { it.participantId == participantId }
    if (index != -1) {
        update(participants[index])
        notifyItemChanged(index, participants[index])
    }
}

```

Kembali `MainViewModel` kita perlu membuat dan menahan referensi ke adaptor ini:

```

internal val participantAdapter = ParticipantAdapter()

```

Status Panggung

Kita juga perlu melacak beberapa status tahap di dalamnya `MainViewModel`. Mari kita definisikan properti itu sekarang:

```

private val _connectionState = MutableStateFlow(Stage.ConnectionState.DISCONNECTED)
val connectionState = _connectionState.asStateFlow()

private var publishEnabled: Boolean = false
    set(value) {

```

```

        field = value
        // Because the strategy returns the value of `checkboxPublish.isChecked`, just
        call `refreshStrategy`.
        stage?.refreshStrategy()
    }

private var deviceDiscovery: DeviceDiscovery? = null
private var stage: Stage? = null
private var streams = mutableListOf<LocalStageStream>()

```

Untuk melihat pratinjau Anda sendiri sebelum bergabung dengan panggung, kami segera membuat peserta lokal:

```

init {
    deviceDiscovery = DeviceDiscovery(application)

    // Create a local participant immediately to render our camera preview and
    microphone stats
    val localParticipant = StageParticipant(true, null)
    participantAdapter.participantJoined(localParticipant)
}

```

Kami ingin memastikan kami membersihkan sumber daya ini ketika kami `ViewModel` dibersihkan. Kami `onCleared()` segera mengganti, jadi kami tidak lupa untuk membersihkan sumber daya ini.

```

override fun onCleared() {
    stage?.release()
    deviceDiscovery?.release()
    deviceDiscovery = null
    super.onCleared()
}

```

Sekarang kita mengisi `streams` properti lokal kita segera setelah izin diberikan, menerapkan `permissionsGranted` metode yang kita sebut sebelumnya:

```

internal fun permissionGranted() {
    val deviceDiscovery = deviceDiscovery ?: return
    streams.clear()
    val devices = deviceDiscovery.listLocalDevices()
    // Camera
    devices
        .filter { it.descriptor.type == Device.Descriptor.DeviceType.CAMERA }
}

```

```

        .maxByOrNull { it.descriptor.position == Device.Descriptor.Position.FRONT }
        ?.let { streams.add(ImageLocalStageStream(it)) }
// Microphone
devices
    .filter { it.descriptor.type == Device.Descriptor.DeviceType.MICROPHONE }
    .maxByOrNull { it.descriptor.isDefault }
    ?.let { streams.add(AudioLocalStageStream(it)) }

stage?.refreshStrategy()

// Update our local participant with these new streams
participantAdapter.participantUpdated(null) {
    it.streams.clear()
    it.streams.addAll(streams)
}
}

```

Menerapkan Stage SDK

Tiga [konsep](#) inti mendasari fungsionalitas real-time: panggung, strategi, dan penyaji. Tujuan desain adalah meminimalkan jumlah logika sisi klien yang diperlukan untuk membangun produk yang berfungsi.

Tahap.Strategi

Stage.StrategyImplementasi kami sederhana:

```

override fun stageStreamsToPublishForParticipant(
    stage: Stage,
    participantInfo: ParticipantInfo
): MutableList<LocalStageStream> {
    // Return the camera and microphone to be published.
    // This is only called if `shouldPublishFromParticipant` returns true.
    return streams
}

override fun shouldPublishFromParticipant(stage: Stage, participantInfo:
ParticipantInfo): Boolean {
    return publishEnabled
}

override fun shouldSubscribeToParticipant(stage: Stage, participantInfo:
ParticipantInfo): Stage.SubscribeType {

```



```
// Subscribe to both audio and video for all publishing participants.  
return Stage.SubscribeType.AUDIO_VIDEO  
}
```

Untuk meringkas, kami menerbitkan berdasarkan `publishEnabled` keadaan internal kami, dan jika kami mempublikasikan, kami akan mempublikasikan aliran yang kami kumpulkan sebelumnya. Akhirnya untuk sampel ini, kami selalu berlangganan peserta lain, menerima audio dan video mereka.

StageRenderer

`StageRenderer` implementasinya juga cukup sederhana, meskipun mengingat jumlah fungsi yang berisi lebih banyak kode. Pendekatan umum dalam penyaji ini adalah memperbarui `ParticipantAdapter` saat SDK memberi tahu kami tentang perubahan pada peserta. Ada skenario tertentu di mana kami menangani peserta lokal secara berbeda, karena kami telah memutuskan untuk mengelolanya sendiri sehingga mereka dapat melihat pratinjau kamera mereka sebelum bergabung.

```
override fun onError(exception: BroadcastException) {  
    Toast.makeText(getApplication(), "onError ${exception.localizedMessage}",  
        Toast.LENGTH_LONG).show()  
    Log.e("BasicRealTime", "onError $exception")  
}  
  
override fun onConnectionStateChanged(  
    stage: Stage,  
    connectionState: Stage.ConnectionState,  
    exception: BroadcastException?  
) {  
    _connectionState.value = connectionState  
}  
  
override fun onParticipantJoined(stage: Stage, participantInfo: ParticipantInfo) {  
    if (participantInfo.isLocal) {  
        // If this is the local participant joining the stage, update the participant  
        with a null ID because we  
        // manually added that participant when setting up our preview  
        participantAdapter.participantUpdated(null) {  
            it.participantId = participantInfo.participantId  
        }  
    } else {  
        // If they are not local, add them normally  
        participantAdapter.participantJoined(  

```

```
        StageParticipant(
            participantInfo.isLocal,
            participantInfo.participantId
        )
    )
}

override fun onParticipantLeft(stage: Stage, participantInfo: ParticipantInfo) {
    if (participantInfo.isLocal) {
        // If this is the local participant leaving the stage, update the ID but keep
        it around because
        // we want to keep the camera preview active
        participantAdapter.participantUpdated(participantInfo.participantId) {
            it.participantId = null
        }
    } else {
        // If they are not local, have them leave normally
        participantAdapter.participantLeft(participantInfo.participantId)
    }
}

override fun onParticipantPublishStateChanged(
    stage: Stage,
    participantInfo: ParticipantInfo,
    publishState: Stage.PublishState
) {
    // Update the publishing state of this participant
    participantAdapter.participantUpdated(participantInfo.participantId) {
        it.publishState = publishState
    }
}

override fun onParticipantSubscribeStateChanged(
    stage: Stage,
    participantInfo: ParticipantInfo,
    subscribeState: Stage.SubscribeState
) {
    // Update the subscribe state of this participant
    participantAdapter.participantUpdated(participantInfo.participantId) {
        it.subscribeState = subscribeState
    }
}
```

```
override fun onStreamsAdded(stage: Stage, participantInfo: ParticipantInfo, streams:
MutableList<StageStream>) {
    // We don't want to take any action for the local participant because we track
those streams locally
    if (participantInfo.isLocal) {
        return
    }
    // For remote participants, add these new streams to that participant's streams
array.
    participantAdapter.participantUpdated(participantInfo.participantId) {
        it.streams.addAll(streams)
    }
}

override fun onStreamsRemoved(stage: Stage, participantInfo: ParticipantInfo, streams:
MutableList<StageStream>) {
    // We don't want to take any action for the local participant because we track
those streams locally
    if (participantInfo.isLocal) {
        return
    }
    // For remote participants, remove these streams from that participant's streams
array.
    participantAdapter.participantUpdated(participantInfo.participantId) {
        it.streams.removeAll(streams)
    }
}

override fun onStreamsMutedChanged(
    stage: Stage,
    participantInfo: ParticipantInfo,
    streams: MutableList<StageStream>
) {
    // We don't want to take any action for the local participant because we track
those streams locally
    if (participantInfo.isLocal) {
        return
    }
    // For remote participants, notify the adapter that the participant has been
updated. There is no need to modify
    // the `streams` property on the `StageParticipant` because it is the same
`StageStream` instance. Just
    // query the `isMuted` property again.
    participantAdapter.participantUpdated(participantInfo.participantId) {}
}
```

}

Menerapkan Kustom RecyclerView LayoutManager

Menempatkan jumlah peserta yang berbeda bisa menjadi rumit. Anda ingin mereka mengambil seluruh bingkai tampilan induk tetapi Anda tidak ingin menangani setiap konfigurasi peserta secara independen. Untuk membuatnya mudah, kita akan berjalan melalui penerapan `aRecyclerView.LayoutManager`.

Buat kelas baru lainnya, `StageLayoutManager`, yang harus diperluas `GridLayoutManager`. Kelas ini dirancang untuk menghitung tata letak untuk setiap peserta berdasarkan jumlah peserta dalam tata letak baris/kolom berbasis aliran. Setiap baris memiliki tinggi yang sama dengan yang lain, tetapi kolom dapat memiliki lebar yang berbeda per baris. Lihat komentar kode di atas `layouts` variabel untuk deskripsi tentang cara menyesuaikan perilaku ini.

```
package com.amazonaws.ivs.realtime.basicrealtime

import android.content.Context
import androidx.recyclerview.widget.GridLayoutManager
import androidx.recyclerview.widget.RecyclerView

class StageLayoutManager(context: Context?) : GridLayoutManager(context, 6) {

    companion object {
        /**
         * This 2D array contains the description of how the grid of participants
         should be rendered
         * The index of the 1st dimension is the number of participants needed to
         active that configuration
         * Meaning if there is 1 participant, index 0 will be used. If there are 5
         participants, index 4 will be used.
         *
         * The 2nd dimension is a description of the layout. The length of the array is
         the number of rows that
         * will exist, and then each number within that array is the number of columns
         in each row.
         *
         * See the code comments next to each index for concrete examples.
         *
         * This can be customized to fit any layout configuration needed.
         */
        val layouts: List<List<Int>> = listOf(
```

```

        // 1 participant
        listOf(1), // 1 row, full width
        // 2 participants
        listOf(1, 1), // 2 rows, all columns are full width
        // 3 participants
        listOf(1, 2), // 2 rows, first row's column is full width then 2nd row's
columns are 1/2 width
        // 4 participants
        listOf(2, 2), // 2 rows, all columns are 1/2 width
        // 5 participants
        listOf(1, 2, 2), // 3 rows, first row's column is full width, 2nd and 3rd
row's columns are 1/2 width
        // 6 participants
        listOf(2, 2, 2), // 3 rows, all column are 1/2 width
        // 7 participants
        listOf(2, 2, 3), // 3 rows, 1st and 2nd row's columns are 1/2 width, 3rd
row's columns are 1/3rd width
        // 8 participants
        listOf(2, 3, 3),
        // 9 participants
        listOf(3, 3, 3),
        // 10 participants
        listOf(2, 3, 2, 3),
        // 11 participants
        listOf(2, 3, 3, 3),
        // 12 participants
        listOf(3, 3, 3, 3),
    )
}

init {
    spanSizeLookup = object : SpanSizeLookup() {
        override fun getSpanSize(position: Int): Int {
            if (itemCount <= 0) {
                return 1
            }
            // Calculate the row we're in
            val config = layouts[itemCount - 1]
            var row = 0
            var currentPosition = position
            while (currentPosition - config[row] >= 0) {
                currentPosition -= config[row]
                row++
            }
        }
    }
}

```

```

        // spanCount == max spans, config[row] = number of columns we want
        // So spanCount / config[row] would be something like 6 / 3 if we want
3 columns.
        // So this will take up 2 spans, with a max of 6 is 1/3rd of the view.
        return spanCount / config[row]
    }
}
}

override fun onLayoutChildren(recycler: RecyclerView.Recycler?, state:
RecyclerView.State?) {
    if (itemCount <= 0 || state?.isPreLayout == true) return

    val parentHeight = height
    val itemHeight = parentHeight / layouts[itemCount - 1].size // height divided
by number of rows.

    // Set the height of each view based on how many rows exist for the current
participant count.
    for (i in 0 until childCount) {
        val child = getChildAt(i) ?: continue
        val layoutParams = child.layoutParams as RecyclerView.LayoutParams
        if (layoutParams.height != itemHeight) {
            layoutParams.height = itemHeight
            child.layoutParams = layoutParams
        }
    }
    // After we set the height for all our views, call super.
    // This works because our RecyclerView can not scroll and all views are always
visible with stable IDs.
    super.onLayoutChildren(recycler, state)
}

override fun canScrollVertically(): Boolean = false
override fun canScrollHorizontally(): Boolean = false
}

```

Kembali MainActivity.kt kita perlu mengatur adaptor dan pengelola tata letak untuk RecyclerView:

```

// In onCreate after setting recyclerView.
recyclerView.layoutManager = StageLayoutManager(this)
recyclerView.adapter = viewModel.participantAdapter

```

Menghubungkan Tindakan UI

Kami semakin dekat; hanya ada beberapa tindakan UI yang perlu kita hubungkan.

Pertama kita akan MainActivity mengamati StateFlow perubahan dariMainViewModel:

```
// At the end of your onCreate method
lifecycleScope.launch {
    repeatOnLifecycle(Lifecycle.State.CREATED) {
        viewModel.connectionState.collect { state ->
            buttonJoin.setText(if (state == ConnectionState.DISCONNECTED) R.string.join
            else R.string.leave)
            textViewState.text = getString(R.string.state, state.name)
        }
    }
}
```

Selanjutnya kita menambahkan pendengar ke tombol Join dan kotak centang Publish kami:

```
buttonJoin.setOnClickListener {
    viewModel.joinStage(editTextToken.text.toString())
}
checkboxPublish.setOnCheckedChangeListener { _, isChecked ->
    viewModel.setPublishEnabled(isChecked)
}
```

Kedua fungsi panggilan di atas di kamiMainViewModel, yang kami terapkan sekarang:

```
internal fun joinStage(token: String) {
    if (_connectionState.value != Stage.ConnectionState.DISCONNECTED) {
        // If we're already connected to a stage, leave it.
        stage?.leave()
    } else {
        if (token.isEmpty()) {
            Toast.makeText(getApplication(), "Empty Token", Toast.LENGTH_SHORT).show()
            return
        }
        try {
            // Destroy the old stage first before creating a new one.
            stage?.release()
            val stage = Stage(getApplication(), token, this)
            stage.addRenderer(this)
            stage.join()
        }
    }
}
```

```
        this.stage = stage
    } catch (e: BroadcastException) {
        Toast.makeText(getApplication(), "Failed to join stage
${e.localizedMessage}", Toast.LENGTH_LONG).show()
        e.printStackTrace()
    }
}

internal fun setPublishEnabled(enabled: Boolean) {
    publishEnabled = enabled
}
```

Rendering Peserta

Akhirnya, kita perlu merender data yang kita terima dari SDK ke item peserta yang kita buat sebelumnya. Kami sudah menyelesaikan RecyclerView logika, jadi kami hanya perlu mengimplementasikan bind API diParticipantItem.

Kita akan mulai dengan menambahkan fungsi kosong dan kemudian berjalan melalui langkah demi langkah:

```
fun bind(participant: StageParticipant) {
}
```

Pertama kita akan menangani status mudah, ID peserta, status publikasi, dan status berlangganan. Untuk ini, kami hanya memperbarui kami TextViews secara langsung:

```
val participantId = if (participant.isLocal) {
    "You (${participant.participantId ?: "Disconnected"})"
} else {
    participant.participantId
}
textViewParticipantId.text = participantId
textViewPublish.text = participant.publishState.name
textViewSubscribe.text = participant.subscribeState.name
```

Selanjutnya kita akan memperbarui status audio dan video yang diredam. Untuk mendapatkan status yang diredam, kita perlu menemukan ImageDevice dan AudioDevice dari array stream. Untuk mengoptimalkan kinerja, kami mengingat ID perangkat terakhir yang terpasang.


```
// This belongs outside the `bind` API.
private var imageDeviceUrn: String? = null
private var audioDeviceUrn: String? = null

// This belongs inside the `bind` API.
val newImageStream = participant
    .streams
    .firstOrNull { it.device is ImageDevice }
textViewVideoMuted.text = if (newImageStream != null) {
    if (newImageStream.muted) "Video muted" else "Video not muted"
} else {
    "No video stream"
}

val newAudioStream = participant
    .streams
    .firstOrNull { it.device is AudioDevice }
textViewAudioMuted.text = if (newAudioStream != null) {
    if (newAudioStream.muted) "Audio muted" else "Audio not muted"
} else {
    "No audio stream"
}
```

Akhirnya kami ingin membuat pratinjau untuk `imageDevice`:

```
if (newImageStream?.device?.descriptor?.urn != imageDeviceUrn) {
    // If the device has changed, remove all subviews from the preview container
    previewContainer.removeAllViews()
    (newImageStream?.device as? ImageDevice)?.let {
        val preview = it.getPreviewView(BroadcastConfiguration.AspectMode.FIT)
        previewContainer.addView(preview)
        preview.layoutParams = FrameLayout.LayoutParams(
            FrameLayout.LayoutParams.MATCH_PARENT,
            FrameLayout.LayoutParams.MATCH_PARENT
        )
    }
}
imageDeviceUrn = newImageStream?.device?.descriptor?.urn
```

Dan kami menampilkan statistik audio dari: `audioDevice`

```
if (newAudioStream?.device?.descriptor?.urn != audioDeviceUrn) {
```

```
(newAudioStream?.device as? AudioDevice)?.let {
    it.setStatsCallback { _, rms ->
        textViewAudioLevel.text = "Audio Level: ${rms.roundToInt()} dB"
    }
}
audioDeviceUrn = newAudioStream?.device?.descriptor?.urn
```

iOS

Buat Tampilan

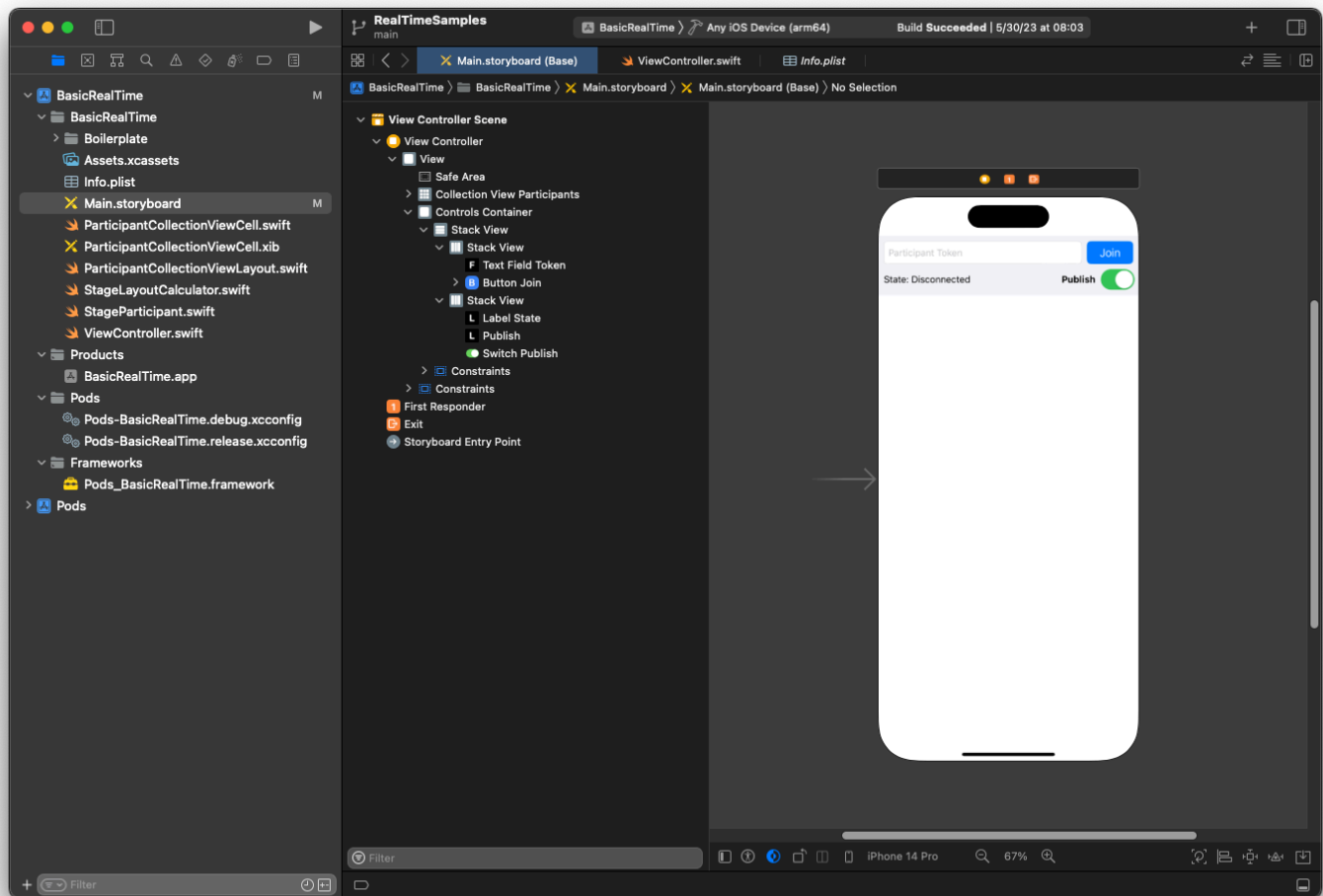
Kami mulai dengan menggunakan `ViewController.swift` file yang dibuat secara otomatis untuk mengimpor `AmazonIVSBroadcast` dan kemudian menambahkan beberapa `@IBOutlet` ke tautan:

```
import AmazonIVSBroadcast

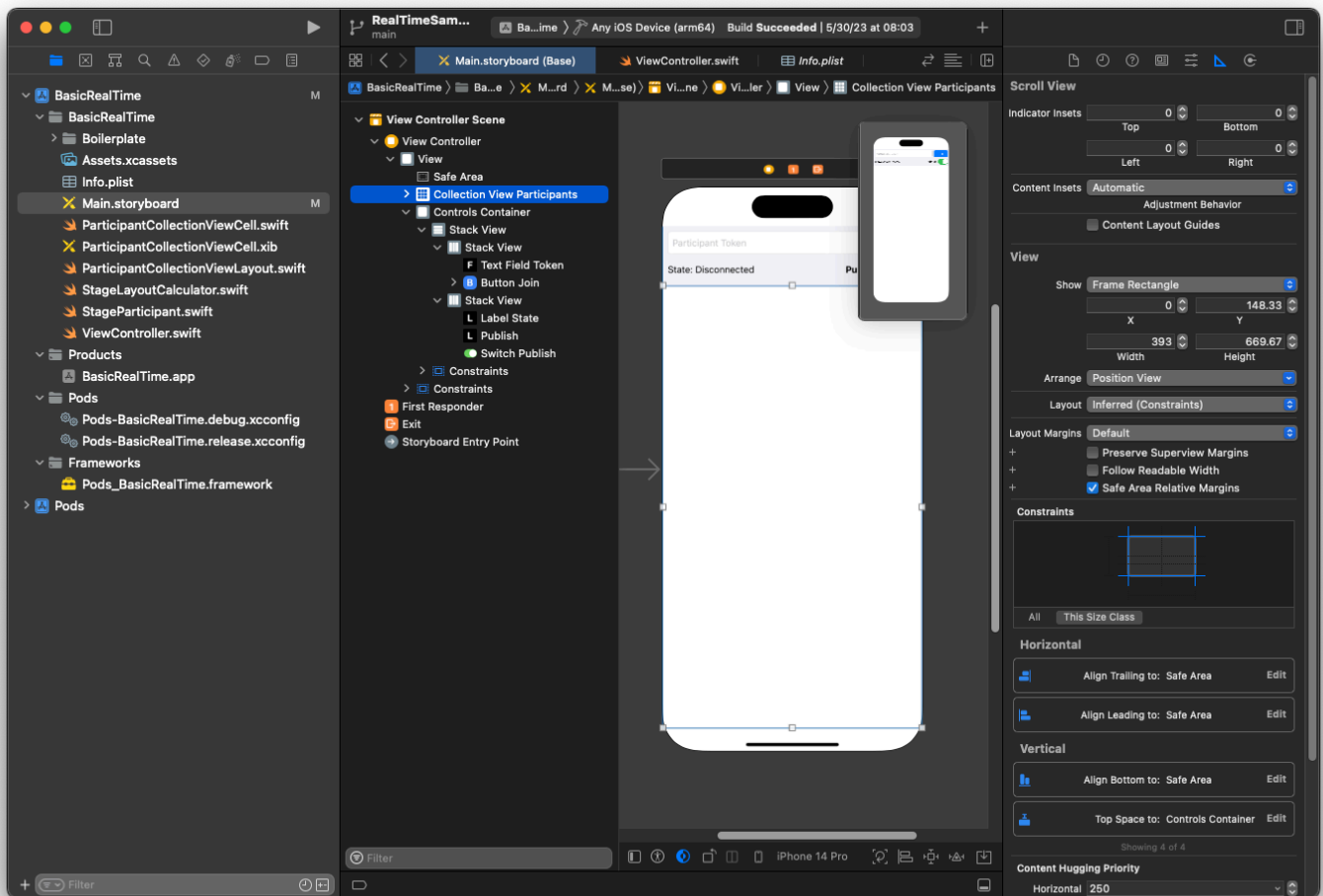
class ViewController: UIViewController {

    @IBOutlet private var textFieldToken: UITextField!
    @IBOutlet private var buttonJoin: UIButton!
    @IBOutlet private var labelState: UILabel!
    @IBOutlet private var switchPublish: UISwitch!
    @IBOutlet private var collectionViewParticipants: UICollectionView!
```

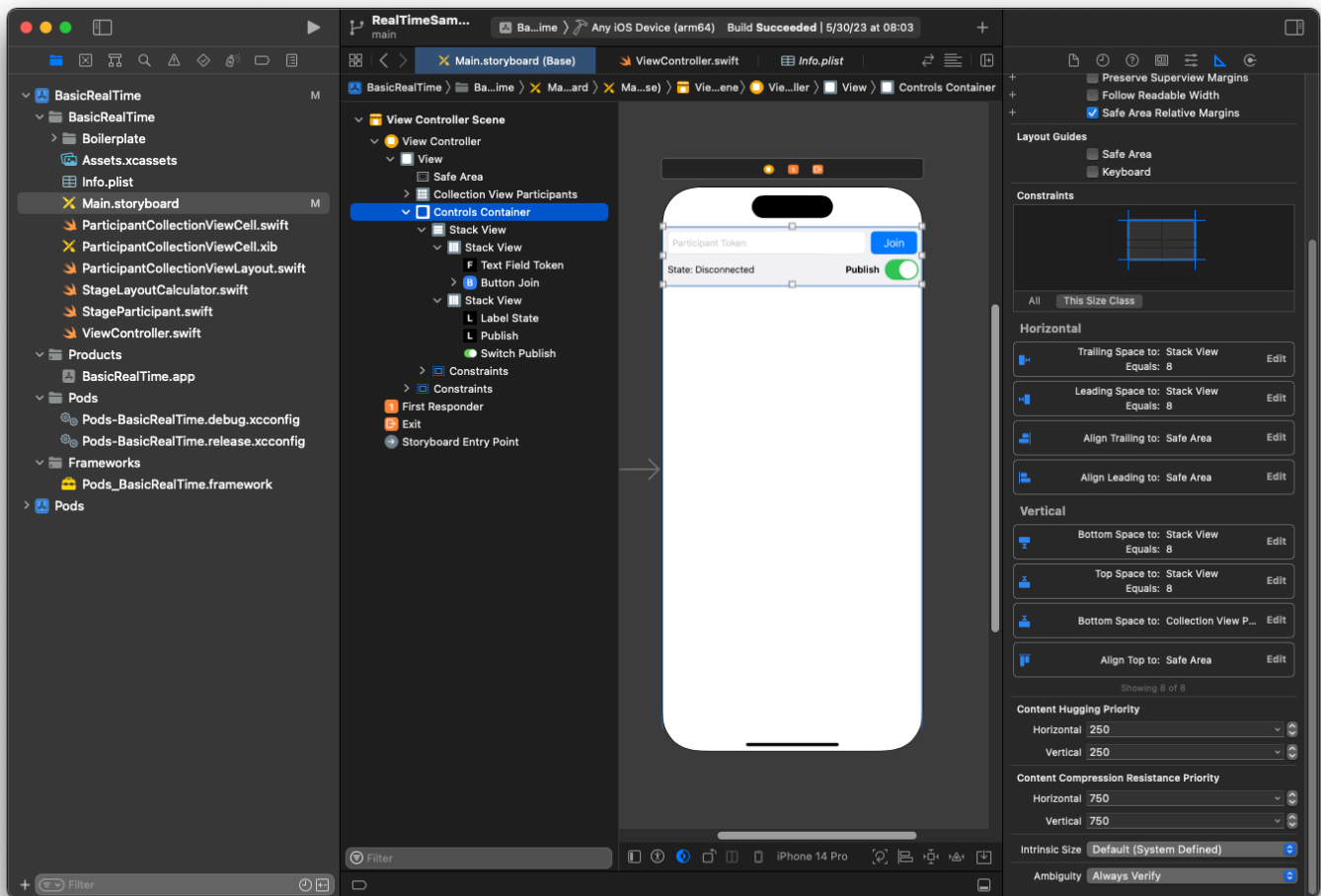
Sekarang kita membuat pandangan itu dan menghubungkannya `Main.storyboard`. Berikut adalah struktur tampilan yang akan kita gunakan:



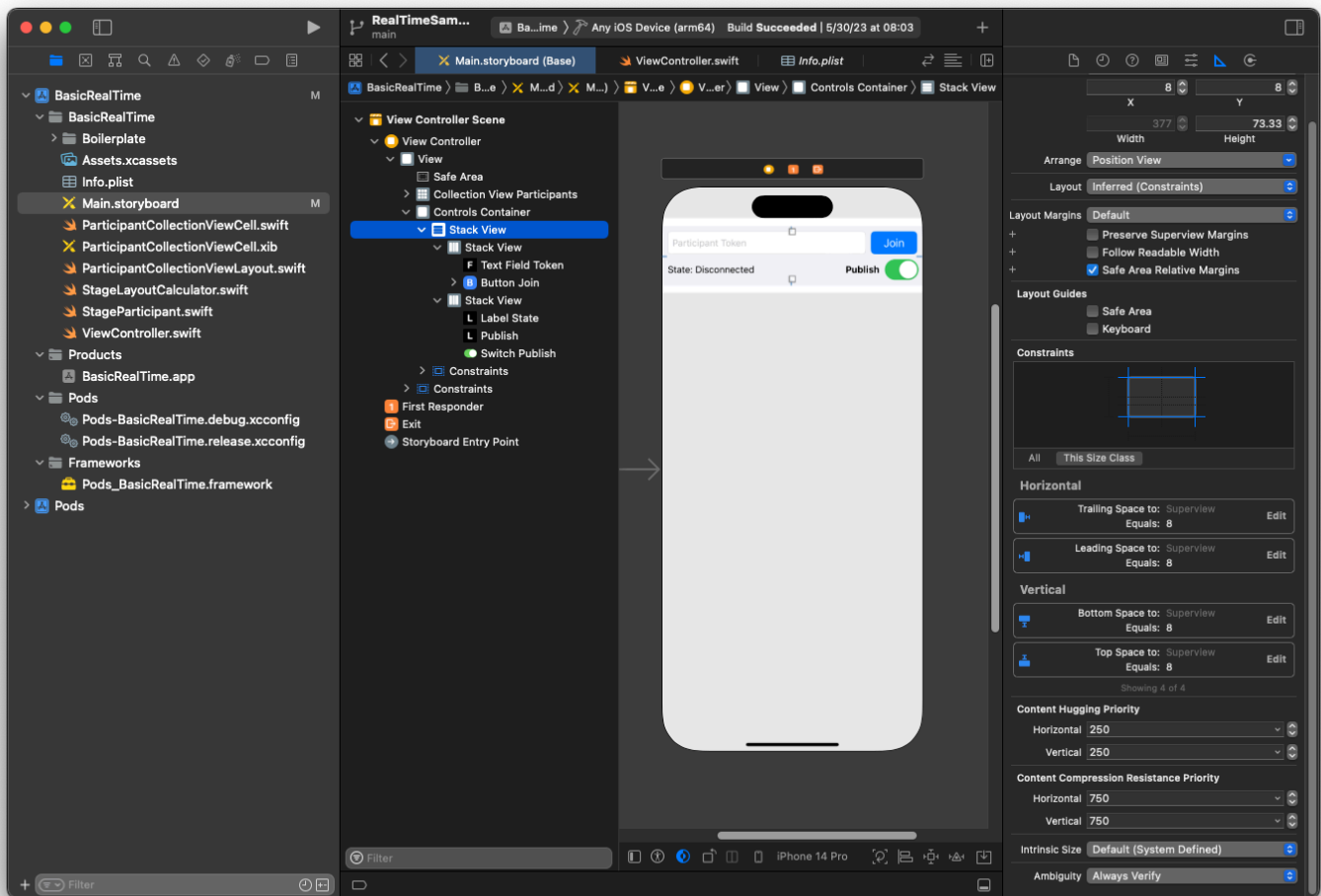
Untuk AutoLayout konfigurasi, kita perlu menyesuaikan tiga tampilan. Tampilan pertama adalah Collection View Partisipan (`UICollectionView`). Terikat Memimpin, Trailing, dan Bawah ke Area Aman. Juga terikat Atas ke Controls Container.



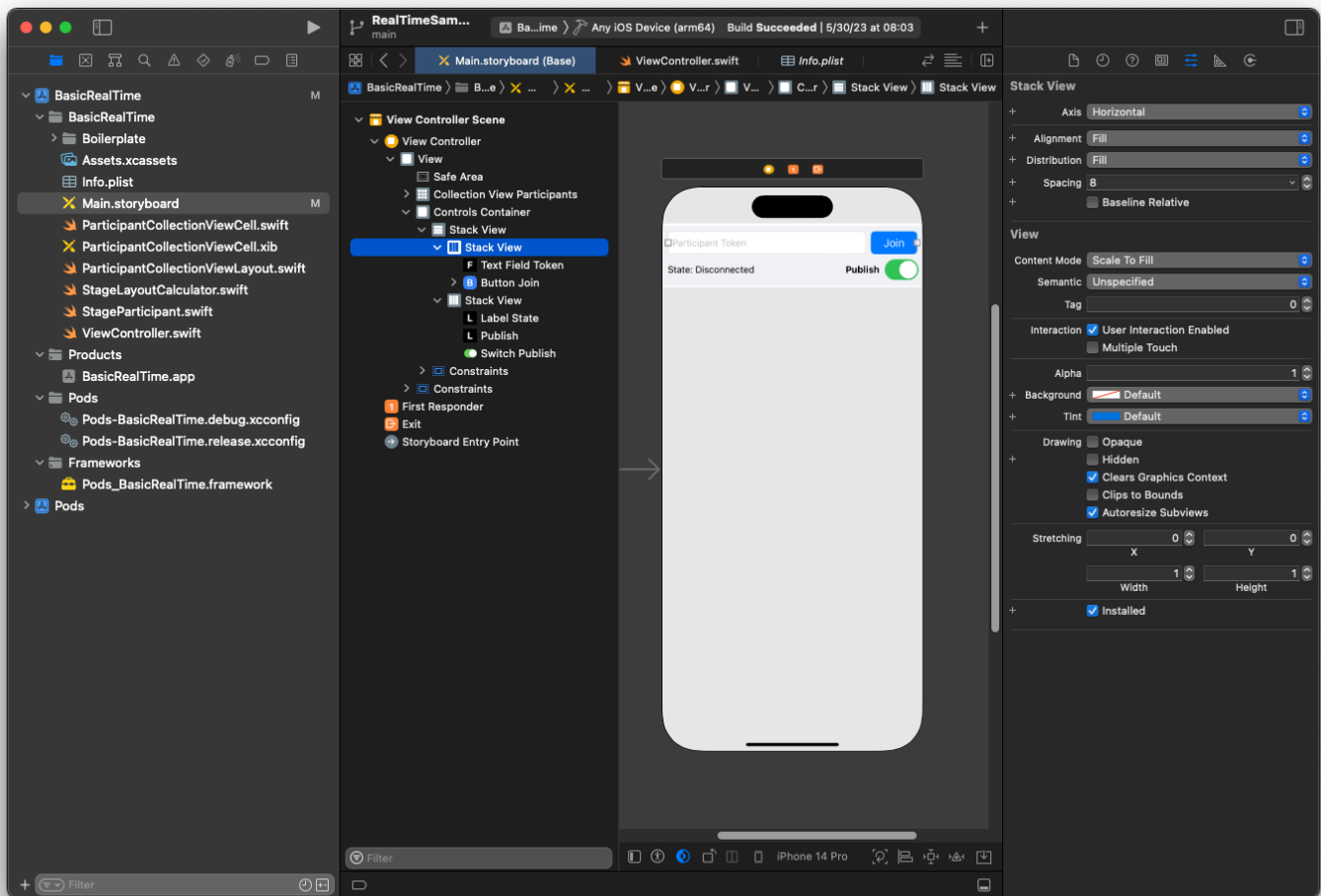
Tampilan kedua adalah Controls Container. Terikat Memimpin, Melintasi, dan Atas ke Area Aman:



Tampilan ketiga dan terakhir adalah Vertical Stack View. Bound Top, Leading, Trailing, dan Bottom ke Superview. Untuk styling, atur spasi ke 8, bukan 0.



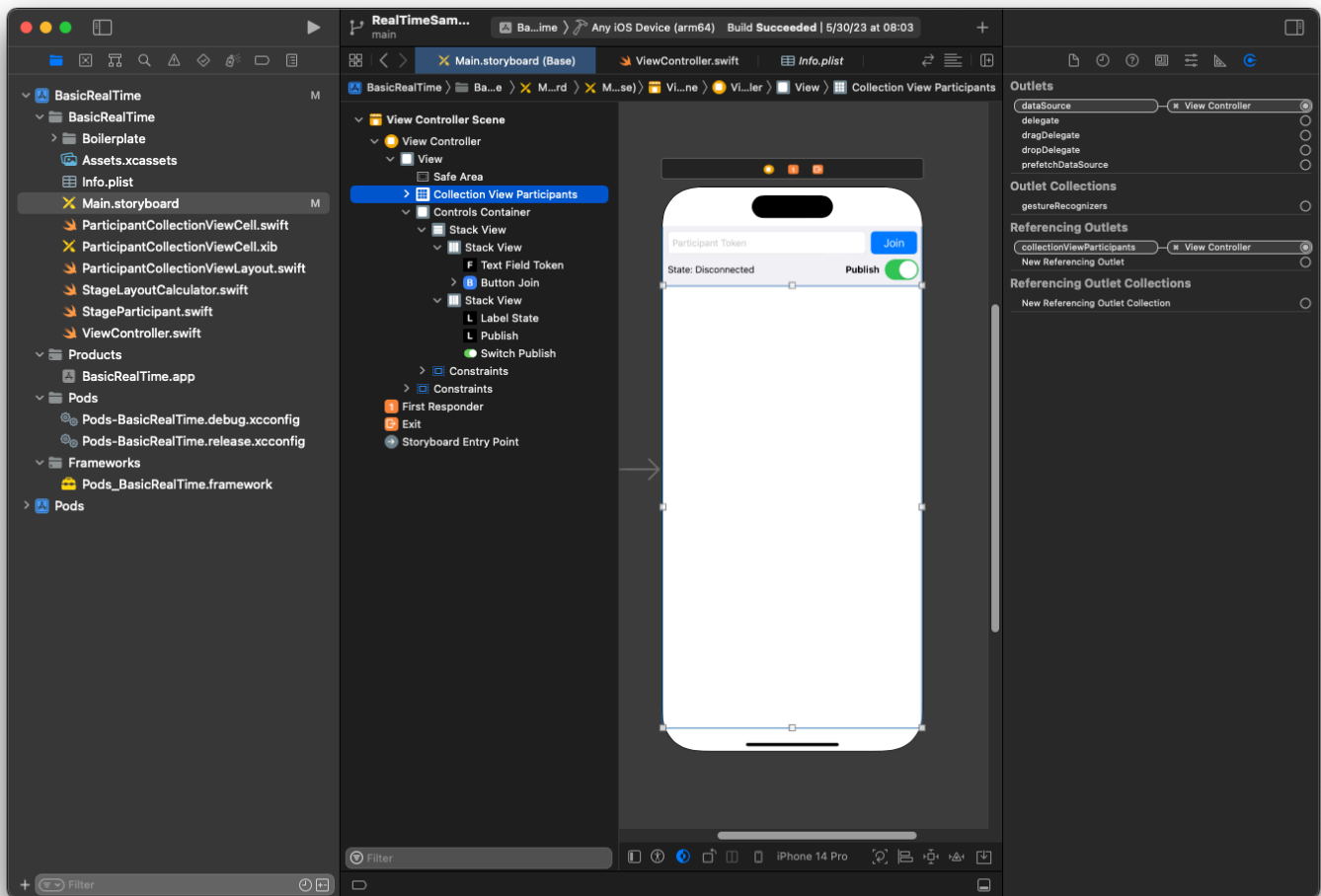
UI StackViews akan menangani tata letak tampilan yang tersisa. Untuk ketiga UI StackViews, gunakan Fill sebagai Alignment and Distribution.



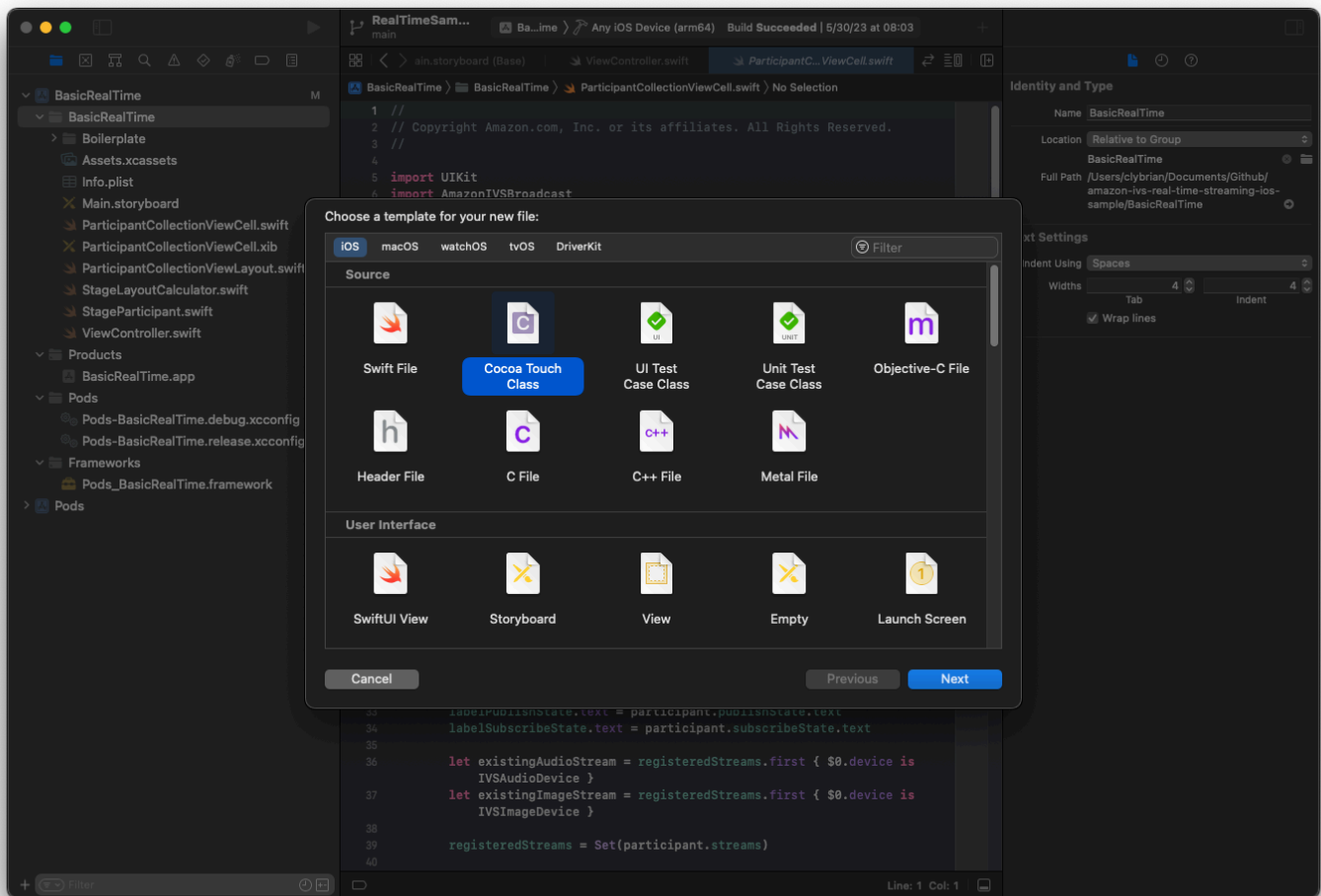
Akhirnya, mari kita tautkan pandangan ini ke pandangan kita `ViewController`. Dari atas, petakan tampilan berikut:

- Bidang Teks Gabung mengikat `textFieldToken`.
- Tombol Gabung mengikat `buttonJoin`.
- Label State mengikat `labelState`.
- Beralih Publikasikan mengikat `switchPublish`.
- Tampilan Koleksi Peserta mengikat `collectionViewParticipants`.

Gunakan juga waktu ini untuk mengatur item Peserta Tampilan Koleksi menjadi milik `ViewController`: `dataSource`



Sekarang kita membuat UICollectionViewCell subclass untuk membuat peserta. Mulailah dengan membuat file Cocoa Touch Class baru:



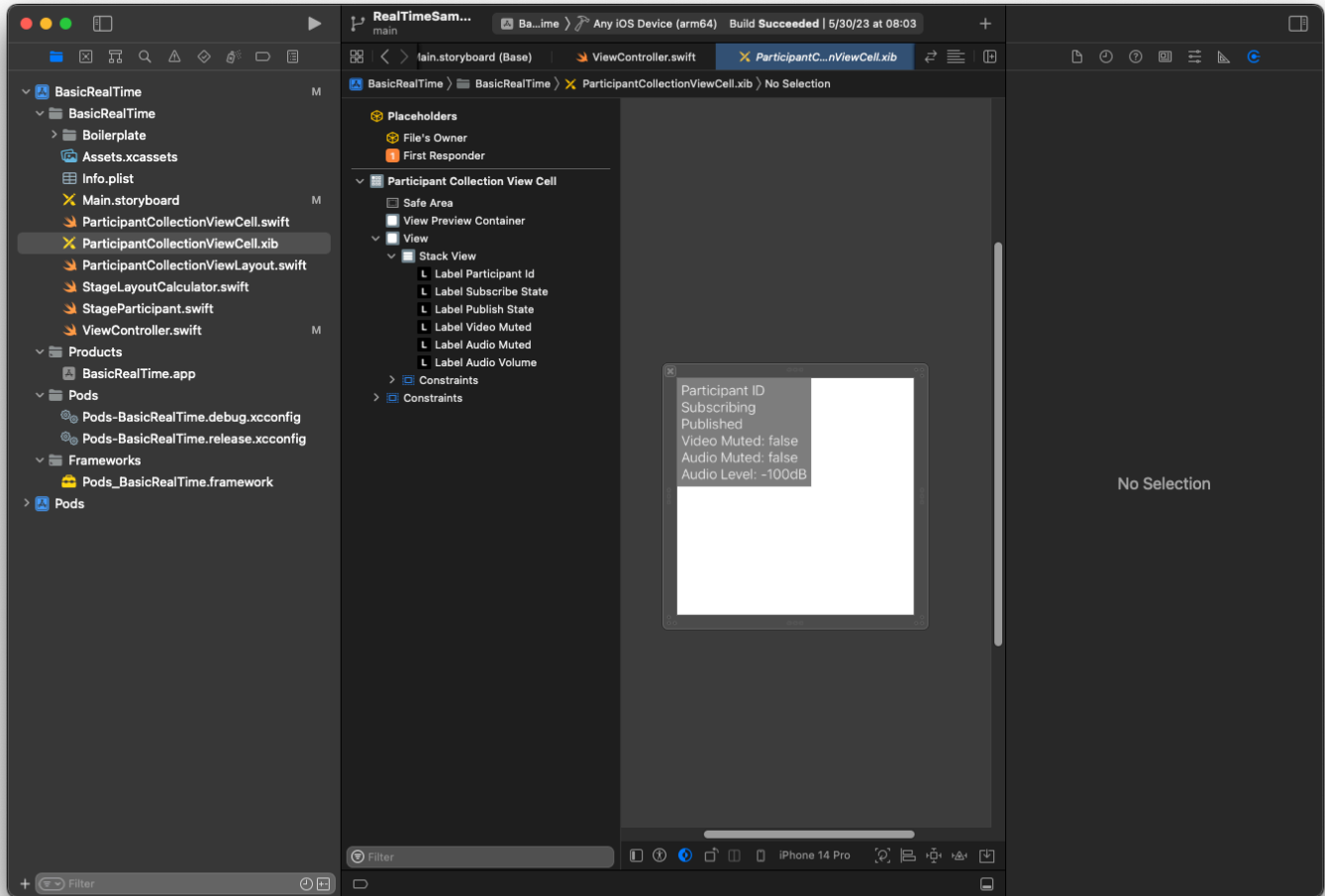
Beri nama ParticipantUICollectionViewCell dan jadikan subclass UICollectionViewCell di Swift. Kami mulai di file Swift lagi, membuat tautan @IBOutlets untuk kami:

```
import AmazonIVSBroadcast

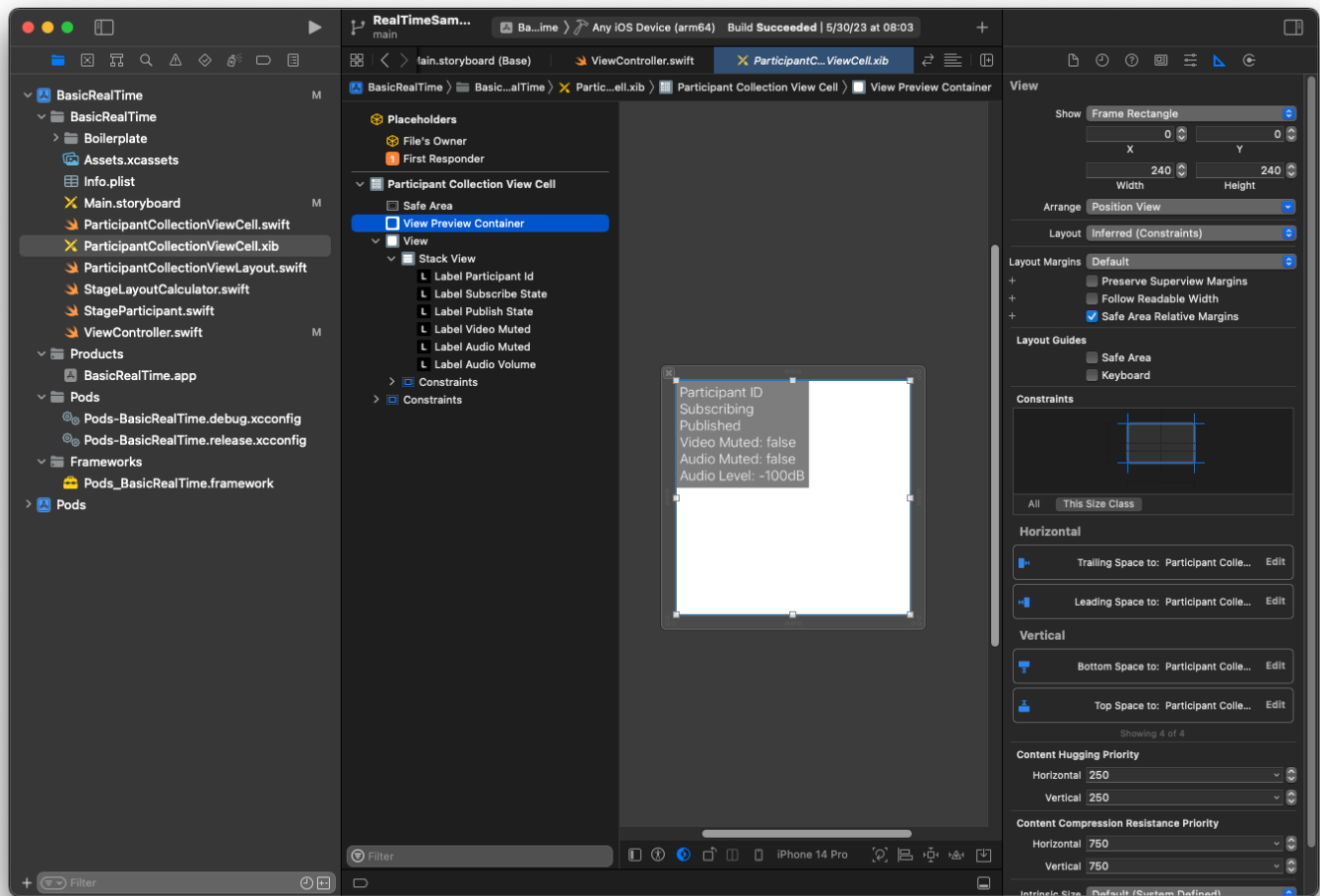
class ParticipantCollectionViewCell: UICollectionViewCell {

    @IBOutlet private var viewPreviewContainer: UIView!
    @IBOutlet private var labelParticipantId: UILabel!
    @IBOutlet private var labelSubscribeState: UILabel!
    @IBOutlet private var labelPublishState: UILabel!
    @IBOutlet private var labelVideoMuted: UILabel!
    @IBOutlet private var labelAudioMuted: UILabel!
    @IBOutlet private var labelAudioVolume: UILabel!
```

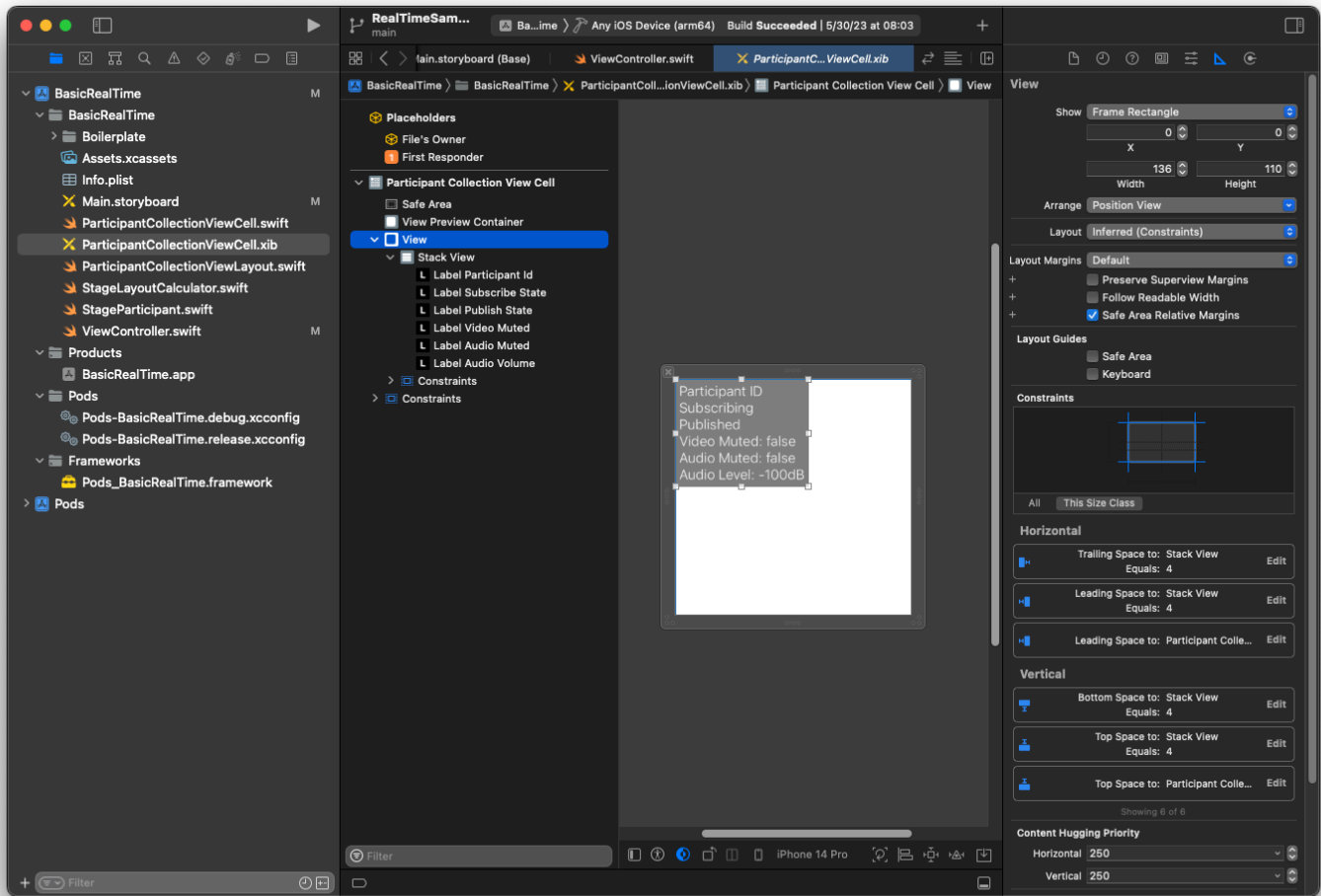
Dalam file XIB terkait, buat hierarki tampilan ini:



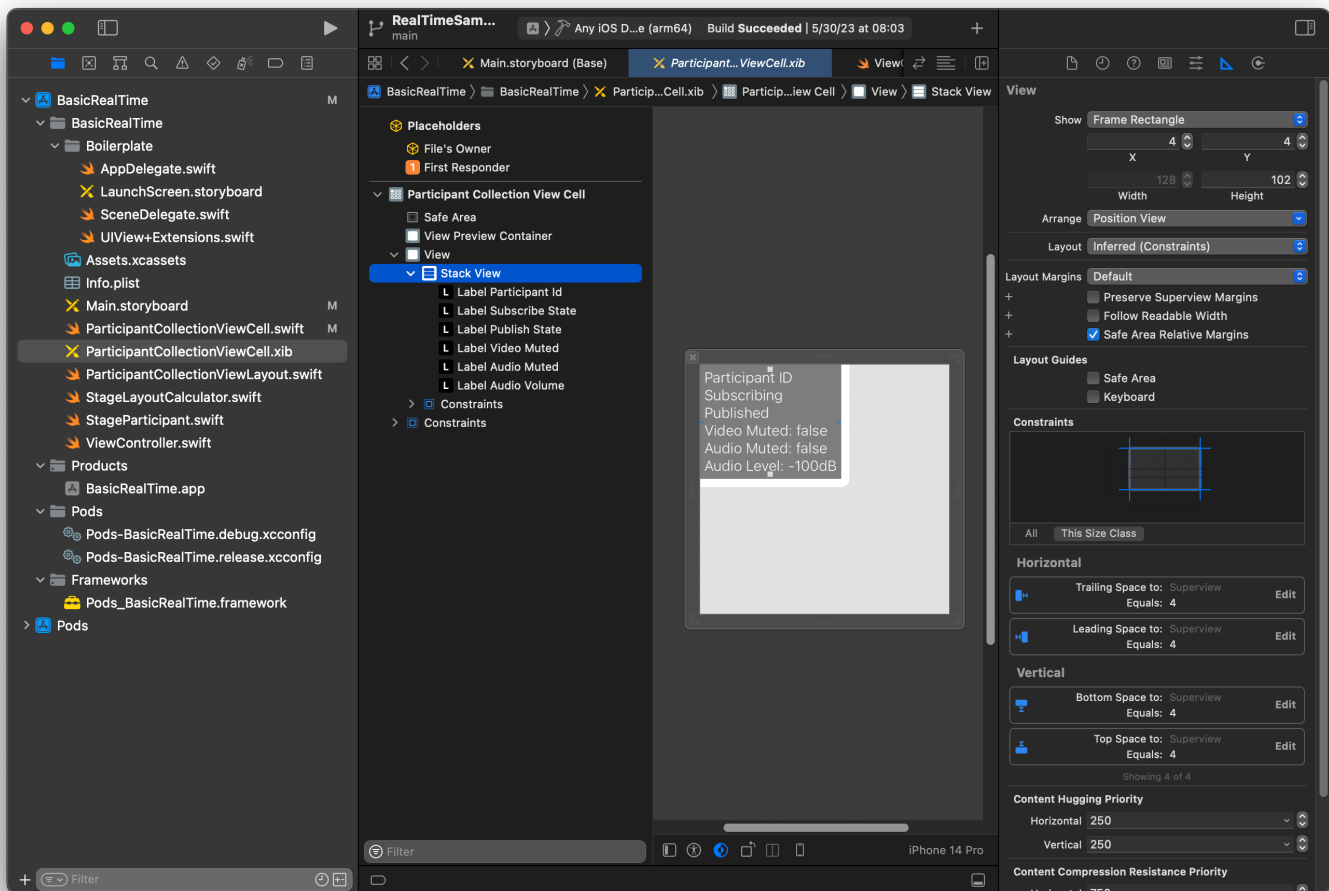
Untuk AutoLayout, kita akan memodifikasi tiga tampilan lagi. Tampilan pertama adalah View Preview Container. Atur Trailing, Leading, Top, dan Bottom ke Sel Tampilan Koleksi Peserta.



Pandangan kedua adalah View. Atur Leading dan Top ke Sel Tampilan Koleksi Peserta dan ubah nilainya menjadi 4.



Tampilan ketiga adalah Stack View. Atur Trailing, Leading, Top, dan Bottom ke Superview dan ubah nilainya menjadi 4.



Izin dan Timer Idle

Kembali ke `kamiViewController`, kami akan menonaktifkan pengatur waktu idle sistem untuk mencegah perangkat tidur saat aplikasi kami digunakan:

```

override func viewWillAppear(_ animated: Bool) {
    super.viewWillAppear(animated)
    // Prevent the screen from turning off during a call.
    UIApplication.shared.isIdleTimerDisabled = true
}

override func viewWillDisappear(_ animated: Bool) {
    super.viewWillDisappear(animated)
    UIApplication.shared.isIdleTimerDisabled = false
}

```

Selanjutnya kami meminta izin kamera dan mikrofon dari sistem:

```

private func checkPermissions() {
    checkOrGetPermission(for: .video) { [weak self] granted in
        guard granted else {
            print("Video permission denied")
            return
        }
        self?.checkOrGetPermission(for: .audio) { [weak self] granted in
            guard granted else {
                print("Audio permission denied")
                return
            }
            self?.setupLocalUser() // we will cover this later
        }
    }
}

private func checkOrGetPermission(for mediaType: AVMediaType, _ result: @escaping
(Bool) -> Void) {
    func mainThreadResult(_ success: Bool) {
        DispatchQueue.main.async {
            result(success)
        }
    }
    switch AVCaptureDevice.authorizationStatus(for: mediaType) {
    case .authorized: mainThreadResult(true)
    case .notDetermined:
        AVCaptureDevice.requestAccess(for: mediaType) { granted in
            mainThreadResult(granted)
        }
    case .denied, .restricted: mainThreadResult(false)
    @unknown default: mainThreadResult(false)
    }
}

```

Status Aplikasi

Kita perlu mengkonfigurasi kita `collectionViewParticipants` dengan file tata letak yang kita buat sebelumnya:

```

override func viewDidLoad() {
    super.viewDidLoad()
    // We render everything to exactly the frame, so don't allow scrolling.
    collectionViewParticipants.isScrollEnabled = false
}

```

```
collectionViewParticipants.register(UINib(nibName: "ParticipantCollectionViewCell",
bundle: .main), forCellWithReuseIdentifier: "ParticipantCollectionViewCell")
}
```

Untuk mewakili setiap peserta, kita membuat struct sederhana yang disebut `StageParticipant`. Ini dapat dimasukkan dalam `ViewController.swift` file, atau file baru dapat dibuat.

```
import Foundation
import AmazonIVSBroadcast

struct StageParticipant {
    let isLocal: Bool
    var participantId: String?
    var publishState: IVSParticipantPublishState = .notPublished
    var subscribeState: IVSParticipantSubscribeState = .notSubscribed
    var streams: [IVSStageStream] = []

    init(isLocal: Bool, participantId: String?) {
        self.isLocal = isLocal
        self.participantId = participantId
    }
}
```

Untuk melacak peserta tersebut, kami menyimpan sederet dari mereka sebagai milik pribadi `diViewController`:

```
private var participants = [StageParticipant]()
```

Properti ini akan digunakan untuk memberi daya pada kami `UICollectionViewDataSource` yang ditautkan dari storyboard sebelumnya:

```
extension ViewController: UICollectionViewDataSource {

    func collectionView(_ collectionView: UICollectionView, numberOfItemsInSection
section: Int) -> Int {
        return participants.count
    }

    func collectionView(_ collectionView: UICollectionView, cellForItemAt indexPath:
IndexPath) -> UICollectionViewCell {
        if let cell = collectionView.dequeueReusableCell(withReuseIdentifier:
"ParticipantCollectionViewCell", for: indexPath) as? ParticipantCollectionViewCell {
```

```

        cell.set(participant: participants[indexPath.row])
        return cell
    } else {
        fatalError("Couldn't load custom cell type
'ParticipantCollectionViewCell'")
    }
}
}
}

```

Untuk melihat pratinjau Anda sendiri sebelum bergabung dengan panggung, kami segera membuat peserta lokal:

```

override func viewDidLoad() {
    /* existing UICollectionView code */
    participants.append(StageParticipant(isLocal: true, participantId: nil))
}

```

Ini menghasilkan sel peserta segera dirender setelah aplikasi berjalan, mewakili peserta lokal.

Pengguna ingin dapat melihat diri mereka sendiri sebelum bergabung dengan tahap, jadi selanjutnya kita menerapkan `setupLocalUser()` metode yang dipanggil dari kode penanganan izin sebelumnya. Kami menyimpan referensi kamera dan mikrofon sebagai `IVSLocalStageStream` objek.

```

private var streams = [IVSLocalStageStream]()
private let deviceDiscovery = IVSDeviceDiscovery()

private func setupLocalUser() {
    // Gather our camera and microphone once permissions have been granted
    let devices = deviceDiscovery.listLocalDevices()
    streams.removeAll()
    if let camera = devices.compactMap({ $0 as? IVSCamera }).first {
        streams.append(IVSLocalStageStream(device: camera))
        // Use a front camera if available.
        if let frontSource = camera.listAvailableInputSources().first(where:
{ $0.position == .front }) {
            camera.setPreferredInputSource(frontSource)
        }
    }
    if let mic = devices.compactMap({ $0 as? IVSMicrophone }).first {
        streams.append(IVSLocalStageStream(device: mic))
    }
}

```



```
    }  
    participants[0].streams = streams  
    participantsChanged(index: 0, changeType: .updated)  
}
```

Di sini kami telah menemukan kamera dan mikrofon perangkat melalui SDK dan menyimpannya di `streams` objek lokal kami, kemudian menetapkan `streams` array peserta pertama (peserta lokal yang kami buat sebelumnya) ke `kamistreams`. Akhirnya kami memanggil `participantsChanged` dengan `index 0` dan `changeType` dari `updated`. Fungsi itu adalah fungsi pembantu untuk memperbarui kami `UICollectionView` dengan animasi yang bagus. Begini tampilannya:

```
private func participantsChanged(index: Int, changeType: ChangeType) {  
    switch changeType {  
    case .joined:  
        collectionViewParticipants?.insertItems(at: [IndexPath(item: index, section:  
0)])  
    case .updated:  
        // Instead of doing reloadData, just grab the cell and update it ourselves. It  
saves a create/destroy of a cell  
        // and more importantly fixes some UI flicker. We disable scrolling so the  
index path per cell  
        // never changes.  
        if let cell = collectionViewParticipants?.cellForItem(at: IndexPath(item:  
index, section: 0)) as? ParticipantCollectionViewCell {  
            cell.set(participant: participants[index])  
        }  
    case .left:  
        collectionViewParticipants?.deleteItems(at: [IndexPath(item: index, section:  
0)])  
    }  
}
```

Jangan khawatir `cell.set`; kita akan membahasnya nanti, tapi di situlah kita akan merender konten sel berdasarkan peserta.

`ChangeType` ini adalah enum sederhana:

```
enum ChangeType {  
    case joined, updated, left  
}
```

Akhirnya, kami ingin melacak apakah panggung terhubung. Kami menggunakan sederhana `bool` untuk melacaknya, yang secara otomatis akan memperbarui UI kami ketika diperbarui sendiri.

```
private var connectingOrConnected = false {
    didSet {
        buttonJoin.setTitle(connectingOrConnected ? "Leave" : "Join", for: .normal)
        buttonJoin.tintColor = connectingOrConnected ? .systemRed : .systemBlue
    }
}
```

Menerapkan Stage SDK

Tiga [konsep](#) inti mendasari fungsionalitas real-time: panggung, strategi, dan penyaji. Tujuan desain adalah meminimalkan jumlah logika sisi klien yang diperlukan untuk membangun produk yang berfungsi.

IV StageStrategy

IVSStageStrategyImplementasi kami sederhana:

```
extension ViewController: IVSStageStrategy {
    func stage(_ stage: IVSStage, streamsToPublishForParticipant participant:
IVSParticipantInfo) -> [IVSLocalStageStream] {
        // Return the camera and microphone to be published.
        // This is only called if `shouldPublishParticipant` returns true.
        return streams
    }

    func stage(_ stage: IVSStage, shouldPublishParticipant participant:
IVSParticipantInfo) -> Bool {
        // Our publish status is based directly on the UISwitch view
        return switchPublish.isOn
    }

    func stage(_ stage: IVSStage, shouldSubscribeToParticipant participant:
IVSParticipantInfo) -> IVSStageSubscribeType {
        // Subscribe to both audio and video for all publishing participants.
        return .audioVideo
    }
}
```

Untuk meringkas, kami hanya mempublikasikan jika sakelar publikasi berada di posisi “on”, dan jika kami mempublikasikan, kami akan mempublikasikan aliran yang kami kumpulkan sebelumnya. Akhirnya, untuk sampel ini, kami selalu berlangganan peserta lain, menerima audio dan video mereka.

IV StageRenderer

IVSStageRendererImplementasinya juga cukup sederhana, meskipun mengingat jumlah fungsi yang berisi lebih banyak kode. Pendekatan umum dalam perender ini adalah memperbarui participants array kami saat SDK memberi tahu kami tentang perubahan pada peserta. Ada skenario tertentu di mana kami menangani peserta lokal secara berbeda, karena kami telah memutuskan untuk mengelolanya sendiri sehingga mereka dapat melihat pratinjau kamera mereka sebelum bergabung.

```
extension ViewController: IVSStageRenderer {

    func stage(_ stage: IVSStage, didChange connectionState: IVSStageConnectionState,
withError error: Error?) {
        labelState.text = connectionState.text
        connectingOrConnected = connectionState != .disconnected
    }

    func stage(_ stage: IVSStage, participantDidJoin participant: IVSParticipantInfo) {
        if participant.isLocal {
            // If this is the local participant joining the Stage, update the first
participant in our array because we
            // manually added that participant when setting up our preview
            participants[0].participantId = participant.participantId
            participantsChanged(index: 0, changeType: .updated)
        } else {
            // If they are not local, add them to the array as a newly joined
participant.
            participants.append(StageParticipant(isLocal: false, participantId:
participant.participantId))
            participantsChanged(index: (participants.count - 1), changeType: .joined)
        }
    }

    func stage(_ stage: IVSStage, participantDidLeave participant: IVSParticipantInfo)
{
        if participant.isLocal {
```

```
        // If this is the local participant leaving the Stage, update the first
participant in our array because
        // we want to keep the camera preview active
        participants[0].participantId = nil
        participantsChanged(index: 0, changeType: .updated)
    } else {
        // If they are not local, find their index and remove them from the array.
        if let index = participants.firstIndex(where: { $0.participantId ==
participant.participantId }) {
            participants.remove(at: index)
            participantsChanged(index: index, changeType: .left)
        }
    }
}

func stage(_ stage: IVSStage, participant: IVSParticipantInfo, didChange
publishState: IVSParticipantPublishState) {
    // Update the publishing state of this participant
    mutatingParticipant(participant.participantId) { data in
        data.publishState = publishState
    }
}

func stage(_ stage: IVSStage, participant: IVSParticipantInfo, didChange
subscribeState: IVSParticipantSubscribeState) {
    // Update the subscribe state of this participant
    mutatingParticipant(participant.participantId) { data in
        data.subscribeState = subscribeState
    }
}

func stage(_ stage: IVSStage, participant: IVSParticipantInfo,
didChangeMutedStreams streams: [IVSStageStream]) {
    // We don't want to take any action for the local participant because we track
those streams locally
    if participant.isLocal { return }
    // For remote participants, notify the UICollectionView that they have updated.
There is no need to modify
    // the `streams` property on the `StageParticipant` because it is the same
`IVSStageStream` instance. Just
    // query the `isMuted` property again.
    if let index = participants.firstIndex(where: { $0.participantId ==
participant.participantId }) {
        participantsChanged(index: index, changeType: .updated)
    }
}
```

```

    }
}

func stage(_ stage: IVSStage, participant: IVSParticipantInfo, didAdd streams:
[IVSStageStream]) {
    // We don't want to take any action for the local participant because we track
those streams locally
    if participant.isLocal { return }
    // For remote participants, add these new streams to that participant's streams
array.
    mutatingParticipant(participant.participantId) { data in
        data.streams.append(contentsOf: streams)
    }
}

func stage(_ stage: IVSStage, participant: IVSParticipantInfo, didRemove streams:
[IVSStageStream]) {
    // We don't want to take any action for the local participant because we track
those streams locally
    if participant.isLocal { return }
    // For remote participants, remove these streams from that participant's
streams array.
    mutatingParticipant(participant.participantId) { data in
        let oldUrns = streams.map { $0.device.descriptor().urn }
        data.streams.removeAll(where: { stream in
            return oldUrns.contains(stream.device.descriptor().urn)
        })
    }
}

// A helper function to find a participant by its ID, mutate that participant, and
then update the UICollectionView accordingly.
private func mutatingParticipant(_ participantId: String?, modifier: (inout
StageParticipant) -> Void) {
    guard let index = participants.firstIndex(where: { $0.participantId ==
participantId }) else {
        fatalError("Something is out of sync, investigate if this was a sample app
or SDK issue.")
    }

    var participant = participants[index]
    modifier(&participant)
    participants[index] = participant
    participantsChanged(index: index, changeType: .updated)
}

```

```

    }
}

```

Kode ini menggunakan ekstensi untuk mengubah status koneksi menjadi teks yang ramah manusia:

```

extension IVSStageConnectionState {
    var text: String {
        switch self {
            case .disconnected: return "Disconnected"
            case .connecting: return "Connecting"
            case .connected: return "Connected"
            @unknown default: fatalError()
        }
    }
}
}

```

Menerapkan UI Kustom UICollectionViewLayout

Menempatkan jumlah peserta yang berbeda bisa menjadi rumit. Anda ingin mereka mengambil seluruh bingkai tampilan induk tetapi Anda tidak ingin menangani setiap konfigurasi peserta secara independen. Untuk membuatnya mudah, kita akan berjalan melalui penerapan `UICollectionViewLayout`.

Buat file baru lainnya, `ParticipantCollectionViewLayout.swift`, yang harus diperluas `UICollectionViewLayout`. Kelas ini akan menggunakan kelas lain yang disebut `StageLayoutCalculator`, yang akan kita bahas segera. Kelas menerima nilai bingkai yang dihitung untuk setiap peserta dan kemudian menghasilkan `UICollectionViewLayoutAttributes` objek yang diperlukan.

```

import Foundation
import UIKit

/**
 Code modified from https://developer.apple.com/documentation/uikit/views\_and\_controls/collection\_views/layouts/customizing\_collection\_view\_layouts?language=objc
 */
class ParticipantCollectionViewLayout: UICollectionViewLayout {

    private let layoutCalculator = StageLayoutCalculator()

    private var contentBounds = CGRect.zero
    private var cachedAttributes = [UICollectionViewLayoutAttributes]()

```

```
override func prepare() {
    super.prepare()

    guard let collectionView = collectionView else { return }

    cachedAttributes.removeAll()
    contentBounds = CGRect(origin: .zero, size: collectionView.bounds.size)

    layoutCalculator.calculateFrames(participantCount:
collectionView.numberOfItems(inSection: 0),
                                width: collectionView.bounds.size.width,
                                height: collectionView.bounds.size.height,
                                padding: 4)

    .enumerated()
    .forEach { (index, frame) in
        let attributes = UICollectionViewLayoutAttributes(forCellWith:
IndexPath(item: index, section: 0))
        attributes.frame = frame
        cachedAttributes.append(attributes)
        contentBounds = contentBounds.union(frame)
    }
}

override var collectionViewContentSize: CGSize {
    return contentBounds.size
}

override func shouldInvalidateLayout(forBoundsChange newBounds: CGRect) -> Bool {
    guard let collectionView = collectionView else { return false }
    return !newBounds.size.equalTo(collectionView.bounds.size)
}

override func layoutAttributesForItem(at indexPath: IndexPath) ->
UICollectionViewLayoutAttributes? {
    return cachedAttributes[indexPath.item]
}

override func layoutAttributesForElements(in rect: CGRect) ->
[UICollectionViewLayoutAttributes]? {
    var attributesArray = [UICollectionViewLayoutAttributes]()

    // Find any cell that sits within the query rect.
```

```

        guard let lastIndex = cachedAttributes.indices.last, let firstMatchIndex =
binSearch(rect, start: 0, end: lastIndex) else {
            return attributesArray
        }

        // Starting from the match, loop up and down through the array until all the
attributes
        // have been added within the query rect.
        for attributes in cachedAttributes[..<firstMatchIndex].reversed() {
            guard attributes.frame.maxY >= rect.minY else { break }
            attributesArray.append(attributes)
        }

        for attributes in cachedAttributes[firstMatchIndex...] {
            guard attributes.frame.minY <= rect.maxY else { break }
            attributesArray.append(attributes)
        }

        return attributesArray
    }

    // Perform a binary search on the cached attributes array.
    func binSearch(_ rect: CGRect, start: Int, end: Int) -> Int? {
        if end < start { return nil }

        let mid = (start + end) / 2
        let attr = cachedAttributes[mid]

        if attr.frame.intersects(rect) {
            return mid
        } else {
            if attr.frame.maxY < rect.minY {
                return binSearch(rect, start: (mid + 1), end: end)
            } else {
                return binSearch(rect, start: start, end: (mid - 1))
            }
        }
    }
}

```

Yang lebih penting adalah `StageLayoutCalculator.swift` kelas. Ini dirancang untuk menghitung frame untuk setiap peserta berdasarkan jumlah peserta dalam tata letak baris/kolom berbasis aliran. Setiap baris memiliki tinggi yang sama dengan yang lain, tetapi kolom dapat memiliki

lebar yang berbeda per baris. Lihat komentar kode di atas layouts variabel untuk deskripsi tentang cara menyesuaikan perilaku ini.

```
import Foundation
import UIKit

class StageLayoutCalculator {

    /// This 2D array contains the description of how the grid of participants should
    be rendered
    /// The index of the 1st dimension is the number of participants needed to active
    that configuration
    /// Meaning if there is 1 participant, index 0 will be used. If there are 5
    participants, index 4 will be used.
    ///
    /// The 2nd dimension is a description of the layout. The length of the array is
    the number of rows that
    /// will exist, and then each number within that array is the number of columns in
    each row.
    ///
    /// See the code comments next to each index for concrete examples.
    ///
    /// This can be customized to fit any layout configuration needed.
    private let layouts: [[Int]] = [
        // 1 participant
        [ 1 ], // 1 row, full width
        // 2 participants
        [ 1, 1 ], // 2 rows, all columns are full width
        // 3 participants
        [ 1, 2 ], // 2 rows, first row's column is full width then 2nd row's columns
        are 1/2 width
        // 4 participants
        [ 2, 2 ], // 2 rows, all columns are 1/2 width
        // 5 participants
        [ 1, 2, 2 ], // 3 rows, first row's column is full width, 2nd and 3rd row's
        columns are 1/2 width
        // 6 participants
        [ 2, 2, 2 ], // 3 rows, all column are 1/2 width
        // 7 participants
        [ 2, 2, 3 ], // 3 rows, 1st and 2nd row's columns are 1/2 width, 3rd row's
        columns are 1/3rd width
        // 8 participants
        [ 2, 3, 3 ],
```

```

    // 9 participants
    [ 3, 3, 3 ],
    // 10 participants
    [ 2, 3, 2, 3 ],
    // 11 participants
    [ 2, 3, 3, 3 ],
    // 12 participants
    [ 3, 3, 3, 3 ],
]

// Given a frame (this could be for a UICollectionView, or a Broadcast Mixer's
// canvas), calculate the frames for each
// participant, with optional padding.
func calculateFrames(participantCount: Int, width: CGFloat, height: CGFloat,
padding: CGFloat) -> [CGRect] {
    if participantCount > layouts.count {
        fatalError("Only \(layouts.count) participants are supported at this time")
    }
    if participantCount == 0 {
        return []
    }
    var currentIndex = 0
    var lastFrame: CGRect = .zero

    // If the height is less than the width, the rows and columns will be flipped.
    // Meaning for 6 participants, there will be 2 rows of 3 columns each.
    let isVertical = height > width

    let halfPadding = padding / 2.0

    let layout = layouts[participantCount - 1] // 1 participant is in index 0, so
    '-1`.
    let rowHeight = (isVertical ? height : width) / CGFloat(layout.count)

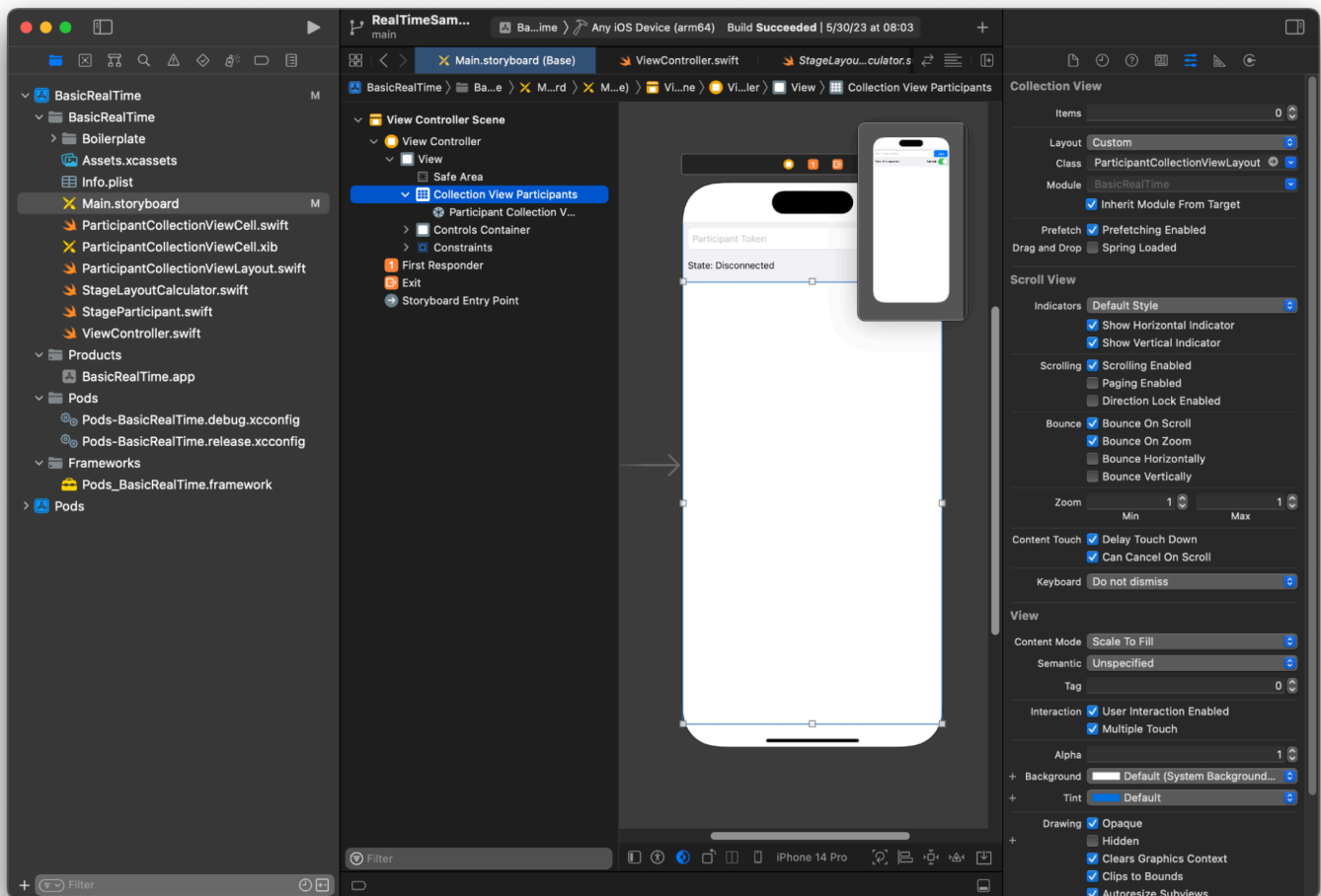
    var frames = [CGRect]()
    for row in 0 ..< layout.count {
        // layout[row] is the number of columns in a layout
        let itemWidth = (isVertical ? width : height) / CGFloat(layout[row])
        let segmentFrame = CGRect(x: (isVertical ? 0 : lastFrame.maxX) +
halfPadding,
                                y: (isVertical ? lastFrame.maxY : 0) +
halfPadding,
                                width: (isVertical ? itemWidth : rowHeight) -
padding,
```

```
padding)
    height: (isVertical ? rowHeight : itemWidth) -

    for column in 0 ..< layout[row] {
        var frame = segmentFrame
        if isVertical {
            frame.origin.x = (itemWidth * CGFloat(column)) + halfPadding
        } else {
            frame.origin.y = (itemWidth * CGFloat(column)) + halfPadding
        }
        frames.append(frame)
        currentIndex += 1
    }

    lastFrame = segmentFrame
    lastFrame.origin.x += halfPadding
    lastFrame.origin.y += halfPadding
}
return frames
}
}
```

KembaliMain.storyboard, pastikan untuk mengatur kelas tata letak untuk kelas yang baru saja kita buat: UICollectionView



Menghubungkan Tindakan UI

Kami semakin dekat, ada beberapa IBActions yang perlu kami buat.

Pertama kita akan menangani tombol join. Ini merespons secara berbeda berdasarkan nilai `connectingOrConnected`. Ketika sudah terhubung, itu hanya meninggalkan panggung. Jika terputus, ia membaca teks dari token `UITextField` dan membuat yang baru `IVSStage` dengan teks itu. Kemudian kita menambahkan kita `ViewController` sebagai `strategy`, `errorDelegate`, dan `renderer` untuk `IVSStage`, dan akhirnya kita bergabung dengan panggung secara asinkron.

```
@IBAction private func joinTapped(_ sender: UIButton) {
    if connectingOrConnected {
        // If we're already connected to a Stage, leave it.
        stage?.leave()
    } else {
        guard let token = textFieldToken.text else {
```

```

        print("No token")
        return
    }
    // Hide the keyboard after tapping Join
    textFieldToken.resignFirstResponder()
    do {
        // Destroy the old Stage first before creating a new one.
        self.stage = nil
        let stage = try IVSStage(token: token, strategy: self)
        stage.errorDelegate = self
        stage.addRenderer(self)
        try stage.join()
        self.stage = stage
    } catch {
        print("Failed to join stage - \(error)")
    }
}
}

```

Tindakan UI lain yang perlu kita hubungkan adalah sakelar publikasikan:

```

@IBAction private func publishToggled(_ sender: UISwitch) {
    // Because the strategy returns the value of `switchPublish.isOn`, just call
    `refreshStrategy`.
    stage?.refreshStrategy()
}

```

Rendering Peserta

Akhirnya, kita perlu merender data yang kita terima dari SDK ke sel peserta yang kita buat sebelumnya. Kami sudah menyelesaikan UICollectionView logika, jadi kami hanya perlu mengimplementasikan set API diParticipantCollectionViewCell.swift.

Kita akan mulai dengan menambahkan empty fungsi dan kemudian berjalan melalui langkah demi langkah:

```

func set(participant: StageParticipant) {
}

```

Pertama kita menangani status mudah, ID peserta, status publikasi, dan status berlangganan. Untuk ini, kami hanya memperbarui kami UILabels secara langsung:

```
labelParticipantId.text = participant.isLocal ? "You (\(participant.participantId ??
"Disconnected"))" : participant.participantId
labelPublishState.text = participant.publishState.text
labelSubscribeState.text = participant.subscribeState.text
```

Properti teks dari mempublikasikan dan berlangganan enum berasal dari ekstensi lokal:

```
extension IVSParticipantPublishState {
    var text: String {
        switch self {
            case .notPublished: return "Not Published"
            case .attemptingPublish: return "Attempting to Publish"
            case .published: return "Published"
            @unknown default: fatalError()
        }
    }
}

extension IVSParticipantSubscribeState {
    var text: String {
        switch self {
            case .notSubscribed: return "Not Subscribed"
            case .attemptingSubscribe: return "Attempting to Subscribe"
            case .subscribed: return "Subscribed"
            @unknown default: fatalError()
        }
    }
}
```

Selanjutnya kami memperbarui status audio dan video yang diredam. Untuk mendapatkan status yang diredam kita perlu menemukan `IVSImageDevice` dan `IVSAudioDevice` dari streams array. Untuk mengoptimalkan kinerja, kami akan mengingat perangkat terakhir yang terpasang.

```
// This belongs outside `set(participant:)`
private var registeredStreams: Set<IVSStageStream> = []
private var imageDevice: IVSImageDevice? {
    return registeredStreams.lazy.compactMap { $0.device as? IVSImageDevice }.first
}
private var audioDevice: IVSAudioDevice? {
    return registeredStreams.lazy.compactMap { $0.device as? IVSAudioDevice }.first
}
```

```
// This belongs inside `set(participant:)`
let existingAudioStream = registeredStreams.first { $0.device is IVSAudioDevice }
let existingImageStream = registeredStreams.first { $0.device is IVSImageDevice }

registeredStreams = Set(participant.streams)

let newAudioStream = participant.streams.first { $0.device is IVSAudioDevice }
let newImageStream = participant.streams.first { $0.device is IVSImageDevice }

// `isMuted != false` covers the stream not existing, as well as being muted.
labelVideoMuted.text = "Video Muted: \(newImageStream?.isMuted != false)"
labelAudioMuted.text = "Audio Muted: \(newAudioStream?.isMuted != false)"
```

Akhirnya kami ingin membuat pratinjau untuk `imageDevice` dan menampilkan statistik audio dari `audioDevice`:

```
if existingImageStream !== newImageStream {
    // The image stream has changed
    updatePreview() // We'll cover this next
}

if existingAudioStream !== newAudioStream {
    (existingAudioStream?.device as? IVSAudioDevice)?.setStatsCallback(nil)
    audioDevice?.setStatsCallback( { [weak self] stats in
        self?.labelAudioVolume.text = String(format: "Audio Level: %.0f dB", stats.rms)
    })
    // When the audio stream changes, it will take some time to receive new stats.
    // Reset the value temporarily.
    self.labelAudioVolume.text = "Audio Level: -100 dB"
}
```

Fungsi terakhir yang perlu kita buat adalah `updatePreview()`, yang menambahkan pratinjau peserta ke tampilan kita:

```
private func updatePreview() {
    // Remove any old previews from the preview container
    viewPreviewContainer.subviews.forEach { $0.removeFromSuperview() }
    if let imageDevice = self.imageDevice {
        if let preview = try? imageDevice.previewView(with: .fit) {
            viewPreviewContainer.addSubviewMatchFrame(preview)
        }
    }
}
```

```
}
```

Di atas menggunakan fungsi pembantu `UIView` untuk mempermudah penyematan subview:

```
extension UIView {
    func addSubviewMatchFrame(_ view: UIView) {
        view.translatesAutoresizingMaskIntoConstraints = false
        self.addSubview(view)
        NSLayoutConstraint.activate([
            view.topAnchor.constraint(equalTo: self.topAnchor, constant: 0),
            view.bottomAnchor.constraint(equalTo: self.bottomAnchor, constant: 0),
            view.leadingAnchor.constraint(equalTo: self.leadingAnchor, constant: 0),
            view.trailingAnchor.constraint(equalTo: self.trailingAnchor, constant: 0),
        ])
    }
}
```


Memantau Streaming Waktu Nyata Amazon IVS

Apa itu Sesi Panggung?

Sesi panggung dimulai ketika peserta pertama bergabung dengan panggung dan berakhir beberapa menit setelah peserta terakhir berhenti menerbitkan ke panggung. Sesi panggung membantu men-debug tahapan berumur panjang dengan memisahkan acara dan peserta menjadi sesi yang berumur pendek.

Lihat Sesi Panggung dan Peserta

Instruksi Konsol

1. Buka [konsol Amazon IVS](#).

(Anda juga dapat mengakses konsol Amazon IVS melalui [Konsol AWS Manajemen](#).)

2. Pada panel navigasi, pilih Tahapan. (Jika panel navigasi dicituk, buka dulu dengan memilih ikon hamburger.)
3. Pilih panggung untuk pergi ke halaman detailnya.
4. Gulir ke bawah halaman hingga Anda melihat bagian Sesi panggung, lalu pilih sesi panggung untuk melihat halaman detailnya.
5. Untuk melihat peserta dalam sesi, gulir ke bawah hingga Anda melihat bagian Peserta, lalu pilih peserta untuk melihat halaman detailnya, termasuk bagan untuk CloudWatch metrik Amazon.

Lihat Acara untuk Peserta

Acara dikirim ketika status peserta dalam tahap berubah, seperti bergabung dengan panggung atau mengalami kesalahan saat mencoba mempublikasikan ke panggung. Tidak semua kesalahan menyebabkan peristiwa; misalnya, kesalahan jaringan sisi klien dan kesalahan tanda tangan token tidak dikirim sebagai peristiwa. Untuk menangani kesalahan ini dalam aplikasi klien Anda, gunakan [SDK siaran IVS](#).

Instruksi Konsol

1. Arahkan ke halaman detail peserta seperti yang diinstruksikan di atas.

2. Gulir ke bawah hingga Anda melihat bagian Acara. Ini menampilkan daftar acara peserta yang diurutkan. Lihat [Menggunakan Amazon EventBridge dengan Amazon IVS](#) untuk detail tentang acara yang dipancarkan untuk peserta.

Instruksi CLI

Mengakses acara sesi panggung dengan AWS CLI adalah opsi lanjutan dan mengharuskan Anda mengunduh dan mengonfigurasi CLI terlebih dahulu di mesin Anda. Untuk detailnya, lihat [Panduan Pengguna Antarmuka Baris AWS Perintah](#).

1. Daftar sesi panggung untuk menemukan sesi panggung:

```
aws ivs-realtime list-stage-sessions --stage-arn <arn>
```

2. Daftar peserta untuk sesi panggung untuk menemukan peserta:

```
aws ivs-realtime list-participants --stage-arn <arn> -session-id <sessionId>
```

3. Daftar acara untuk sesi panggung dan peserta:

```
aws ivs-realtime list-participant-events --stage-arn <arn> --session-id <sessionId> --participant-id <participantId>
```

Berikut adalah contoh respons untuk `list-participant-events` panggilan tersebut:

```
{
  "events": [
    {
      "eventTime": "2023-04-04T22:48:41+00:00",
      "name": "JOINED",
      "participantId": "AdRezB1021t0"
    },
    {
      "eventTime": "2023-04-04T22:48:41+00:00",
      "name": "SUBSCRIBE_STARTED",
      "participantId": "AdRezB1021t0",
      "remoteParticipantId": "0u5b5n5XLMdC"
    },
    {
      "eventTime": "2023-04-04T22:49:45+00:00",
```

```
    "name": "SUBSCRIBE_STOPPED",
    "participantId": "AdRezB1021t0",
    "remoteParticipantId": "0u5b5n5XLMdC"
  },
  {
    "eventTime": "2023-04-04T22:49:45+00:00",
    "name": "LEFT",
    "participantId": "AdRezB1021t0"
  }
]
```

CloudWatchMetrik Akses

Agar CloudWatch metrik tersedia, diperlukan versi IVS Broadcast SDK berikut: Web 1.5.0 atau yang lebih baru, Android 1.12.0 atau versi lebih baru, atau iOS 1.12.0 atau yang lebih baru.

CloudWatch Instruksi Konsol

1. Buka CloudWatch konsol di <https://console.aws.amazon.com/cloudwatch/>.
2. Di navigasi samping, perluas dropdown Metrik, lalu pilih Semua metrik.
3. Pada tab Browse, menggunakan dropdown tidak berlabel di sebelah kiri, pilih wilayah “rumah” Anda, tempat saluran Anda dibuat. Untuk informasi lebih lanjut tentang wilayah, lihat [Solusi Global, Kontrol Regional](#). Untuk daftar wilayah yang didukung, lihat [halaman Amazon IVS](#) di Referensi Umum AWS.
4. Di bagian bawah tab Browse, pilih RealTime namespace IVS.
5. Lakukan salah satu dari berikut:

- a. Di bilah pencarian, masukkan ID sumber daya Anda (bagian dari ARN,`arn:::ivs:stage/<resource id>`).

Kemudian pilih IVS RealTime > Stage Metrics.

- b. Jika IVS RealTime muncul sebagai layanan yang dapat dipilih di bawah AWS Namespaces, pilih layanan tersebut. Ini akan terdaftar jika Anda menggunakan Amazon IVS Real-Time Streaming dan mengirimkan metrik ke Amazon. CloudWatch (Jika IVS RealTime tidak terdaftar, Anda tidak memiliki metrik Amazon IVS.)

Kemudian pilih pengelompokan dimensi sesuai keinginan; dimensi yang tersedia tercantum dalam [CloudWatchMetrik](#) di bawah ini.

6. Pilih metrik untuk ditambahkan ke grafik. Metrik yang tersedia tercantum dalam [CloudWatch Metrik](#) di bawah ini.

Anda juga dapat mengakses CloudWatch bagan sesi streaming Anda dari halaman detail sesi streaming, dengan memilih CloudWatch tombol Lihat di.

Instruksi CLI

Anda juga dapat mengakses metrik menggunakan AWS CLI. Ini mengharuskan Anda mengunduh dan mengonfigurasi CLI terlebih dahulu di mesin Anda. Untuk detailnya, lihat [Panduan Pengguna Antarmuka Baris Perintah AWS](#).

Kemudian, untuk mengakses metrik streaming real-time Amazon IVS menggunakan AWS CLI:

- Pada prompt perintah, jalankan:

```
aws cloudwatch list-metrics --namespace AWS/IVSRealTime
```

Untuk informasi selengkapnya, lihat [Menggunakan CloudWatch Metrik](#) Amazon di Panduan CloudWatch Pengguna Amazon.

CloudWatch Metrik: Streaming Waktu Nyata IVS

Amazon IVS menyediakan metrik berikut di namespace RealTimeAWS/IVS.

Agar CloudWatch metrik tersedia, Web Broadcast SDK 1.5.2 atau yang lebih baru harus digunakan.

Dimensi dapat memiliki nilai valid berikut:

- StageDimensinya adalah ID sumber daya (bagian dari ARN, `arn:::stage/<resource id>`).
- ParticipantDimensinya adalah `aparticipantID`.
- SimulcastLayerItu adalah “hi”, “mid”, “low”, atau “no-rid” untuk “video” atau “dinoaktifkan” untuk “audio.” `Media Type Media Type` Nilai ini juga bisa kosong.
- `Media Type Dimensinya` adalah “video” atau “audio” (string).

Metrik	Dimensi	Deskripsi
DownloadPacketLoss	Stage	<p>Setiap sampel mewakili persentase paket yang hilang oleh pelanggan tertentu saat mengunduh dari server IVS.</p> <p>Satuan: Persen</p> <p>Statistik yang valid: Rata-rata, Maksimum, Minimum — Jumlah rata-rata, jumlah terbesar, atau angka terkecil (masing-masing) dari kehilangan paket selama interval yang dikonfigurasi</p>
DownloadPacketLoss	Stage, Participant	<p>Filter DownloadPacketLoss menurut peserta, untuk pelanggan yang juga penerbit. Sampel mewakili persentase paket yang hilang oleh pelanggan saat mengunduh dari server IVS. Sampel dipancarkan hanya jika peserta juga penerbit.</p> <p>Satuan: Persen</p> <p>Statistik yang valid: Rata-rata, Maksimum, Minimum - Jumlah rata-rata, jumlah terbesar, atau angka terkecil (masing-masing) dari frame yang dijatuhkan selama interval yang dikonfigurasi</p>
DroppedFrames	Stage	<p>Setiap sampel mewakili persentase frame yang dijatuhkan oleh pelanggan tertentu.</p> <p>Satuan: Persen</p> <p>Statistik yang valid: Rata-rata, Maksimum, Minimum - Jumlah rata-rata, jumlah terbesar, atau angka terkecil (masing-masing) dari frame yang dijatuhkan selama interval yang dikonfigurasi</p>
DroppedFrames	Stage, Participant	<p>Filter DroppedFrames menurut peserta, untuk pelanggan yang juga penerbit. Sampel mewakili persentase frame yang dijatuhkan antara peserta</p>

Metrik	Dimensi	Deskripsi
		<p>berlangganan dan semua penerbit di panggung. Sampel dipancarkan hanya jika peserta juga penerbit.</p> <p>Satuan: Persen</p> <p>Statistik yang valid: Rata-rata, Maksimum, Minimum - Jumlah rata-rata, jumlah terbesar, atau angka terkecil (masing-masing) dari frame yang dijatuhkan selama interval yang dikonfigurasi</p>
PublishBitrate	Stage	<p>Sampel yang dipancarkan mewakili tingkat total di mana penerbit tertentu mengirim data video dan audio (dijumlahkan di semua lapisan simulcast).</p> <p>Satuan: Bit per detik</p> <p>Statistik yang valid: Rata-rata, Maksimum, Minimum — Jumlah rata-rata, jumlah terbesar, atau angka terkecil (masing-masing) bitrate selama interval yang dikonfigurasi</p>
PublishBitrate	Stage, Participant, Simulcast Layer, MediaType	<p>Filter PublishBitrate berdasarkan peserta, lapisan simulcast, dan jenis media. ID lapisan simulcast diatur oleh SDK siaran. Ketika simulcast dinonaktifkan, ID lapisan ini akan diatur ke "dininonaktifkan". Jenis media adalah video atau audio.</p> <p>Satuan: Bit per detik</p> <p>Statistik yang valid: Rata-rata, Maksimum, Minimum — Jumlah rata-rata, jumlah terbesar, atau angka terkecil (masing-masing) bitrate selama interval yang dikonfigurasi</p>

Metrik	Dimensi	Deskripsi
Publishers	Stage	<p>Jumlah peserta yang mempublikasikan ke panggung.</p> <p>Unit: Jumlah</p> <p>Statistik yang valid: Rata-rata, Maksimum, Minimum</p>
PublishResolution	Stage, Participant, Simulcast Layer, MediaType	<p>Jumlah piksel di bagian yang lebih kecil dari lebar atau tinggi bingkai. Misalnya, untuk bingkai lanskap ukuran 1920x1080, adalah 1080. PublishResolution Untuk bingkai potret ukuran 720x1280, adalah 720. PublishResolution</p> <p>Unit: Jumlah</p> <p>Statistik yang valid: Rata-rata, Maksimum, Minimum</p>
SubscribeBitrate	Stage	<p>Sampel yang dipancarkan mewakili tingkat total di mana pelanggan tertentu menerima data video dan audio.</p> <p>Satuan: Bit per detik</p> <p>Statistik yang valid: Rata-rata, Maksimum, Minimum — Jumlah rata-rata, jumlah terbesar, atau angka terkecil (masing-masing) bitrate selama interval yang dikonfigurasi</p>
SubscribeBitrate	Stage, Participant, MediaType	<p>Filter <code>SubscribeBitrate</code> menurut peserta, untuk pelanggan yang juga penerbit. Sampel mewakili bitrate di mana pelanggan tertentu menerima yang diberikan. <code>MediaType</code> Sampel hanya dipancarkan saat peserta berlangganan menerbitkan.</p> <p>Satuan: Bit per detik</p> <p>Statistik yang valid: Rata-rata, Maksimum, Minimum — Jumlah rata-rata, jumlah terbesar, atau angka terkecil (masing-masing) bitrate selama interval yang dikonfigurasi</p>

Metrik	Dimensi	Deskripsi
Subscribers	Stage	<p>Jumlah peserta yang berlangganan panggung. Perhatikan bahwa peserta yang secara aktif menerbitkan dan berlangganan dihitung sebagai penerbit dan pelanggan.</p> <p>Unit: Jumlah</p> <p>Statistik yang valid: Rata-rata, Maksimum, Minimum</p>

SDK Siaran IVS (Streaming Waktu Nyata)

Amazon Interactive Video Services (IVS) Real-Time Streaming broadcast SDK adalah untuk pengembang yang sedang membangun aplikasi dengan Amazon IVS. SDK ini dirancang untuk memanfaatkan arsitektur Amazon IVS dan akan melihat peningkatan berkelanjutan dan fitur baru, bersama Amazon IVS. Sebagai SDK siaran asli, SDK ini dirancang untuk meminimalkan dampak kinerja pada aplikasi Anda dan pada perangkat yang digunakan pengguna untuk mengakses aplikasi Anda.

Perhatikan bahwa SDK siaran digunakan untuk mengirim dan menerima video; yaitu, Anda menggunakan SDK yang sama untuk host dan pemirsa. Tidak diperlukan SDK pemain terpisah.

Aplikasi Anda dapat memanfaatkan fitur utama SDK siaran Amazon IVS:

- Streaming berkualitas tinggi - SDK siaran mendukung streaming berkualitas tinggi. Rekam video dari kamera Anda dan encode hingga 720p.
- Penyesuaian Bitrate Otomatis — Pengguna ponsel cerdas bersifat mobile, sehingga kondisi jaringan mereka dapat berubah sepanjang siaran. SDK siaran Amazon IVS secara otomatis menyesuaikan bitrate video untuk mengakomodasi perubahan kondisi jaringan.
- Dukungan Potret dan Lanskap — Tidak peduli bagaimana pengguna Anda memegang perangkat mereka, gambar akan muncul di sisi kanan atas dan diskalakan dengan benar. SDK siaran mendukung ukuran kanvas potret dan lanskap. Ini secara otomatis mengelola rasio aspek ketika pengguna memutar perangkat mereka dari orientasi yang dikonfigurasi.
- Streaming Aman — Siaran pengguna Anda dienkripsi menggunakan TLS, sehingga mereka dapat menjaga aliran mereka tetap aman.
- Perangkat Audio Eksternal - SDK siaran Amazon IVS mendukung jack audio, USB, dan mikrofon eksternal Bluetooth SCO.

Persyaratan Platform

Platform Asli

Platform	Versi yang Didukung
Android	9.0 dan yang lebih baru -- perhatikan pelanggan dapat membangun dengan versi 5.0 tetapi tidak akan dapat menggunakan fungsionalitas streaming waktu nyata.
iOS	14 dan kemudian

IVS mendukung minimal 4 versi iOS utama dan 6 versi Android utama. Dukungan versi kami saat ini dapat melampaui batas minimum ini. Pelanggan akan diberi tahu melalui catatan rilis SDK setidaknya 3 bulan sebelum versi utama tidak lagi didukung.

Browser Desktop

Peramban	Platform yang Didukung	Versi yang Didukung
Chrome	Windows, macOS	Dua versi utama (versi sebelumnya saat ini dan terbaru)
Firefox	Windows, macOS	Dua versi utama (versi sebelumnya saat ini dan terbaru)
Edge	Windows 8.1 dan yang lebih baru	Dua versi utama (versi sebelumnya saat ini dan terbaru) Tidak termasuk Edge Legacy
Safari	macOS	Dua versi utama (versi sebelumnya saat ini dan terbaru)

Browser Seluler (iOS dan Android)

Peramban	Platform yang Didukung	Versi yang Didukung
Chrome	iOS, Android	Dua versi utama (versi sebelumnya saat ini dan terbaru)
Firefox	Android	Dua versi utama (versi sebelumnya saat ini dan terbaru)
Safari	iOS	Dua versi utama (versi sebelumnya saat ini dan terbaru)

Keterbatasan yang Sudah Diketahui

- Di semua perangkat seluler, kami tidak menyarankan penerbitan/berlangganan dengan empat atau lebih peserta secara bersamaan, karena masalah dengan artefak video dan layar hitam. Jika Anda memerlukan lebih banyak peserta, konfigurasi [publikasi dan berlangganan khusus audio](#).
- Kami tidak menyarankan untuk menyusun panggung dan menyiarkannya ke saluran di Android Mobile Web, karena pertimbangan kinerja dan potensi crash. Jika fungsionalitas siaran diperlukan, integrasikan [SDK siaran Android streaming real-time IVS](#).

Tampilan Web

SDK siaran Web tidak menyediakan dukungan untuk tampilan web atau lingkungan seperti web (TV, konsol, dll). [Untuk implementasi seluler, lihat Panduan SDK Siaran Streaming Waktu Nyata untuk Android dan iOS](#).

Akses Perangkat yang Diperlukan

SDK siaran memerlukan akses ke kamera dan mikrofon perangkat, baik yang terpasang di perangkat maupun yang terhubung melalui Bluetooth, USB, atau jack audio.

Dukungan

SDK siaran terus ditingkatkan. Lihat [Catatan Rilis Amazon IVS](#) untuk versi yang tersedia dan masalah yang diperbaiki. Jika perlu, sebelum menghubungi dukungan, perbarui versi SDK siaran Anda dan lihat apakah itu menyelesaikan masalah Anda.

Versioning

SDK siaran Amazon IVS menggunakan versi [semantik](#).

Untuk diskusi ini, misalkan:

- Rilis terbaru adalah 4.1.3.
- Rilis terbaru dari versi utama sebelumnya adalah 3.2.4.
- Rilis terbaru versi 1.x adalah 1.5.6.

Fitur baru yang kompatibel dengan mundur ditambahkan sebagai rilis kecil dari versi terbaru. Dalam hal ini, rangkaian fitur baru berikutnya akan ditambahkan sebagai versi 4.2.0.

Perbaikan bug minor yang kompatibel ke belakang ditambahkan sebagai rilis patch dari versi terbaru. Di sini, set perbaikan bug minor berikutnya akan ditambahkan sebagai versi 4.1.4.

Perbaikan bug utama yang kompatibel ke belakang ditangani secara berbeda; ini ditambahkan ke beberapa versi:

- Rilis patch dari versi terbaru. Di sini, ini adalah versi 4.1.4.
- Rilis patch dari versi minor sebelumnya. Di sini, ini adalah versi 3.2.5.
- Rilis patch dari rilis 1.x versi terbaru. Di sini, ini adalah versi 1.5.7.

Perbaikan bug utama ditentukan oleh tim produk Amazon IVS. Contoh umum adalah pembaruan keamanan penting dan perbaikan lain yang dipilih yang diperlukan untuk pelanggan.

Catatan: Dalam contoh di atas, versi yang dirilis meningkat tanpa melewati angka apa pun (misalnya, dari 4.1.3 ke 4.1.4). Pada kenyataannya, satu atau lebih nomor patch mungkin tetap internal dan tidak dirilis, sehingga versi yang dirilis dapat meningkat dari 4.1.3 menjadi, katakanlah, 4.1.6.

SDK Siaran IVS: Panduan Web (Streaming Waktu Nyata)

IVS real-time streaming Web broadcast SDK memberi pengembang alat untuk membangun pengalaman interaktif dan real-time di web. SDK ini untuk pengembang yang sedang membangun aplikasi web dengan Amazon IVS.

Web broadcast SDK memungkinkan peserta untuk mengirim dan menerima video. SDK mendukung operasi berikut:

- Bergabunglah dengan panggung
- Publikasikan media ke peserta lain di panggung
- Berlangganan media dari peserta lain di panggung
- Kelola dan pantau video dan audio yang dipublikasikan ke panggung
- Dapatkan statistik WebRTC untuk setiap koneksi rekan
- Semua operasi dari SDK siaran Web streaming latensi rendah IVS

Versi terbaru dari Web broadcast SDK: [1.8.0 \(Catatan Rilis\)](#)

Dokumentasi referensi: Untuk informasi tentang metode terpenting yang tersedia di Amazon IVS Web Broadcast SDK, lihat <https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference>. Pastikan versi SDK terbaru dipilih.

Contoh kode: Sampel di bawah ini adalah tempat yang baik untuk memulai dengan cepat dengan SDK:

- [HTML dan JavaScript](#)
- [Bereaksi](#)

Persyaratan platform: Lihat [Amazon IVS Broadcast SDK](#) untuk daftar platform yang didukung

Memulai

Impor

Blok bangunan untuk real-time terletak di namespace yang berbeda dari modul penyiaran root.

Menggunakan Tag Skrip

Menggunakan impor skrip yang sama, kelas dan enum yang didefinisikan dalam contoh di bawah ini dapat ditemukan pada objek global: `IVSBroadcastClient`

```
const { Stage, SubscribeType } = IVSBroadcastClient;
```

Menggunakan npm

Kelas, enum, dan tipe juga dapat diimpor dari modul paket:

```
import { Stage, SubscribeType, LocalStageStream } from 'amazon-ivs-web-broadcast'
```

Permintaan Izin

Aplikasi Anda harus meminta izin untuk mengakses kamera dan mikrofon pengguna, dan harus disajikan menggunakan HTTPS. (Ini tidak khusus untuk Amazon IVS; diperlukan untuk situs web apa pun yang membutuhkan akses ke kamera dan mikrofon.)

Berikut adalah contoh fungsi yang menunjukkan bagaimana Anda dapat meminta dan menangkap izin untuk perangkat audio dan video:

```
async function handlePermissions() {
  let permissions = {
    audio: false,
    video: false,
  };
  try {
    const stream = await navigator.mediaDevices.getUserMedia({ video: true, audio:
true });
    for (const track of stream.getTracks()) {
      track.stop();
    }
    permissions = { video: true, audio: true };
  } catch (err) {
    permissions = { video: false, audio: false };
    console.error(err.message);
  }
  // If we still don't have permissions after requesting them display the error
message
  if (!permissions.video) {
    console.error('Failed to get video permissions.');
```

[Untuk informasi tambahan, lihat API Izin dan `MediaDevices.getUserMedia\(\)`.](#)

Daftar Perangkat yang Tersedia

Untuk melihat perangkat apa yang tersedia untuk ditangkap, kueri metode [MediaDevices.enumerateDevices\(\)](#) browser:

```
const devices = await navigator.mediaDevices.enumerateDevices();
window.videoDevices = devices.filter((d) => d.kind === 'videoinput');
window.audioDevices = devices.filter((d) => d.kind === 'audioinput');
```

Mengambil MediaStream dari Perangkat

Setelah memperoleh daftar perangkat yang tersedia, Anda dapat mengambil aliran dari sejumlah perangkat. Misalnya, Anda dapat menggunakan `getUserMedia()` metode ini untuk mengambil aliran dari kamera.

Jika Anda ingin menentukan perangkat mana yang akan menangkap aliran, Anda dapat secara eksplisit menyetel `video` bagian `audio` atau batasan media. `deviceId` Sebagai alternatif, Anda dapat menghilangkan `deviceId` dan meminta pengguna memilih perangkat mereka dari prompt browser.

Anda juga dapat menentukan resolusi kamera yang ideal menggunakan `width` dan `height` kendala. [\(Baca lebih lanjut tentang kendala ini di sini.\)](#) SDK secara otomatis menerapkan batasan lebar dan tinggi yang sesuai dengan resolusi siaran maksimum Anda; namun, sebaiknya Anda juga menerapkannya sendiri untuk memastikan bahwa rasio aspek sumber tidak berubah setelah Anda menambahkan sumber ke SDK.

Untuk streaming real-time, pastikan media dibatasi hingga resolusi 720p. Secara khusus, nilai Anda `getUserMedia` dan `getDisplayMedia` batasan untuk lebar dan tinggi tidak boleh melebihi 921600 (1280* 720) saat dikalikan bersama.

```
const videoConfiguration = {
  maxWidth: 1280,
  maxHeight: 720,
  maxFramerate: 30,
}

window.cameraStream = await navigator.mediaDevices.getUserMedia({
  video: {
    deviceId: window.videoDevices[0].deviceId,
    width: {
```

```
        ideal: videoConfiguration.maxWidth,
    },
    height: {
        ideal: videoConfiguration.maxHeight,
    },
},
});
window.microphoneStream = await navigator.mediaDevices.getUserMedia({
    audio: { deviceId: window.audioDevices[0].deviceId },
});
```

Menerbitkan dan Berlangganan

Konsep

Tiga konsep inti mendasari fungsionalitas real-time: [panggung](#), [strategi](#), dan [acara](#). Tujuan desain adalah meminimalkan jumlah logika sisi klien yang diperlukan untuk membangun produk yang berfungsi.

Tahap

StageKelas adalah titik utama interaksi antara aplikasi host dan SDK. Ini mewakili panggung itu sendiri dan digunakan untuk bergabung dan meninggalkan panggung. Membuat dan menggabungkan tahap membutuhkan string token yang valid dan belum kedaluwarsa dari bidang kontrol (direpresentasikan sebagai token). Bergabung dan meninggalkan panggung itu sederhana:

```
const stage = new Stage(token, strategy)

try {
    await stage.join();
} catch (error) {
    // handle join exception
}

stage.leave();
```

Strategi

StageStrategyAntarmuka menyediakan cara bagi aplikasi host untuk mengkomunikasikan status tahap yang diinginkan ke SDK. Tiga fungsi perlu

diimplementasikan: `shouldSubscribeToParticipant`, `shouldPublishParticipant`, dan `stageStreamsToPublish`. Semua dibahas di bawah ini.

Untuk menggunakan strategi yang ditentukan, berikan ke Stage konstruktor. Berikut ini adalah contoh lengkap aplikasi yang menggunakan strategi untuk mempublikasikan webcam peserta ke panggung dan berlangganan semua peserta. Setiap tujuan fungsi strategi yang diperlukan dijelaskan secara rinci di bagian selanjutnya.

```
const devices = await navigator.mediaDevices.getUserMedia({
  audio: true,
  video: {
    width: { max: 1280 },
    height: { max: 720 },
  }
});
const myAudioTrack = new LocalStageStream(devices.getAudioTracks()[0]);
const myVideoTrack = new LocalStageStream(devices.getVideoTracks()[0]);

// Define the stage strategy, implementing required functions
const strategy = {
  audioTrack: myAudioTrack,
  videoTrack: myVideoTrack,

  // optional
  updateTracks(newAudioTrack, newVideoTrack) {
    this.audioTrack = newAudioTrack;
    this.videoTrack = newVideoTrack;
  },

  // required
  stageStreamsToPublish() {
    return [this.audioTrack, this.videoTrack];
  },

  // required
  shouldPublishParticipant(participant) {
    return true;
  },

  // required
  shouldSubscribeToParticipant(participant) {
    return SubscribeType.AUDIO_VIDEO;
  }
}
```

```
};

// Initialize the stage and start publishing
const stage = new Stage(token, strategy);
await stage.join();

// To update later (e.g. in an onClick event handler)
strategy.updateTracks(myNewAudioTrack, myNewVideoTrack);
stage.refreshStrategy();
```

Berlangganan Peserta

```
shouldSubscribeToParticipant(participant: StageParticipantInfo): SubscribeType
```

Ketika peserta jarak jauh bergabung dengan panggung, SDK menanyakan aplikasi host tentang status langganan yang diinginkan untuk peserta tersebut. Pilihannya adalah `NONE`, `AUDIO_ONLY`, dan `AUDIO_VIDEO`. Saat mengembalikan nilai untuk fungsi ini, aplikasi host tidak perlu khawatir tentang status publikasi, status langganan saat ini, atau status koneksi tahap. Jika `AUDIO_VIDEO` dikembalikan, SDK menunggu hingga peserta jarak jauh mempublikasikan sebelum berlangganan, dan memperbarui aplikasi host dengan memancarkan peristiwa selama proses berlangsung.

Berikut adalah contoh implementasi:

```
const strategy = {

  shouldSubscribeToParticipant: (participant) => {
    return SubscribeType.AUDIO_VIDEO;
  }

  // ... other strategy functions
}
```

Ini adalah implementasi lengkap dari fungsi ini untuk aplikasi host yang selalu ingin semua peserta saling bertemu; misalnya, aplikasi obrolan video.

Implementasi yang lebih maju juga dimungkinkan. Gunakan `userInfo` properti `ParticipantInfo` untuk berlangganan peserta secara selektif berdasarkan atribut yang disediakan server:

```
const strategy = {
```

```
shouldSubscribeToParticipant(participant) {
  switch (participant.info.userInfo) {
    case 'moderator':
      return SubscribeType.NONE;
    case 'guest':
      return SubscribeType.AUDIO_VIDEO;
    default:
      return SubscribeType.NONE;
  }
}
// . . . other strategies properties
}
```

Ini dapat digunakan untuk membuat panggung di mana moderator dapat memantau semua tamu tanpa terlihat atau didengar sendiri. Aplikasi host dapat menggunakan logika bisnis tambahan untuk membiarkan moderator melihat satu sama lain tetapi tetap tidak terlihat oleh tamu.

Penerbitan

```
shouldPublishParticipant(participant: StageParticipantInfo): boolean
```

Setelah terhubung ke panggung, SDK menanyakan aplikasi host untuk melihat apakah peserta tertentu harus mempublikasikannya. Ini hanya dipanggil pada peserta lokal yang memiliki izin untuk mempublikasikan berdasarkan token yang disediakan.

Berikut adalah contoh implementasi:

```
const strategy = {
  shouldPublishParticipant: (participant) => {
    return true;
  }
  // . . . other strategies properties
}
```

Ini untuk aplikasi obrolan video standar di mana pengguna selalu ingin mempublikasikan. Mereka dapat membisukan dan menonaktifkan audio dan video mereka, untuk langsung disembunyikan atau dilihat/didengar. (Mereka juga dapat menggunakan publish/unpublish, tetapi itu jauh lebih lambat. Mute/unmute lebih disukai untuk kasus penggunaan di mana perubahan visibilitas sering diinginkan.)

Memilih Streaming untuk Publikasikan

```
stageStreamsToPublish(): LocalStageStream[];
```

Saat menerbitkan, ini digunakan untuk menentukan aliran audio dan video apa yang harus dipublikasikan. Ini dibahas secara lebih rinci nanti di [Publish a Media Stream](#).

Memperbarui Strategi

Strategi ini dimaksudkan untuk menjadi dinamis: nilai yang dikembalikan dari salah satu fungsi di atas dapat diubah kapan saja. Misalnya, jika aplikasi host tidak ingin mempublikasikan sampai pengguna akhir mengetuk tombol, Anda dapat mengembalikan variabel dari `shouldPublishParticipant` (sesuatu seperti `hasUserTappedPublishButton`). Ketika variabel itu berubah berdasarkan interaksi oleh pengguna akhir, panggil `stage.refreshStrategy()` untuk memberi sinyal ke SDK bahwa ia harus menanyakan strategi untuk nilai terbaru, hanya menerapkan hal-hal yang telah berubah. Jika SDK mengamati bahwa `shouldPublishParticipant` nilainya telah berubah, SDK memulai proses publikasi. Jika kueri SDK dan semua fungsi mengembalikan nilai yang sama seperti sebelumnya, `refreshStrategy` panggilan tidak mengubah tahapan.

Jika nilai pengembalian `shouldSubscribeToParticipant` perubahan dari `AUDIO_VIDEO` ke `AUDIO_ONLY`, aliran video akan dihapus untuk semua peserta dengan nilai yang dikembalikan diubah, jika aliran video sudah ada sebelumnya.

Umumnya, tahap menggunakan strategi untuk menerapkan perbedaan antara strategi sebelumnya dan saat ini secara efisien, tanpa aplikasi host perlu khawatir tentang semua keadaan yang diperlukan untuk mengelolanya dengan benar. Karena itu, anggap menelepon `stage.refreshStrategy()` sebagai operasi yang murah, karena tidak melakukan apa-apa kecuali strateginya berubah.

Peristiwa

`StageInstance` adalah emitor peristiwa. Menggunakan `stage.on()`, keadaan panggung dikomunikasikan ke aplikasi host. Pembaruan UI aplikasi host biasanya dapat didukung sepenuhnya oleh acara. Peristiwa-peristiwa tersebut adalah sebagai berikut:

```
stage.on(StageEvents.STAGE_CONNECTION_STATE_CHANGED, (state) => {})  
stage.on(StageEvents.STAGE_PARTICIPANT_JOINED, (participant) => {})  
stage.on(StageEvents.STAGE_PARTICIPANT_LEFT, (participant) => {})  
stage.on(StageEvents.STAGE_PARTICIPANT_PUBLISH_STATE_CHANGED, (participant, state) => {})
```

```
stage.on(StageEvents.STAGE_PARTICIPANT_SUBSCRIBE_STATE_CHANGED, (participant, state) =>
  {})
stage.on(StageEvents.STAGE_PARTICIPANT_STREAMS_ADDED, (participant, streams) => {})
stage.on(StageEvents.STAGE_PARTICIPANT_STREAMS_REMOVED, (participant, streams) => {})
stage.on(StageEvents.STAGE_STREAM_MUTE_CHANGED, (participant, stream) => {})
```

Untuk sebagian besar acara ini, yang sesuai `ParticipantInfo` disediakan.

Tidak diharapkan bahwa informasi yang diberikan oleh peristiwa berdampak pada nilai pengembalian strategi. Misalnya, nilai pengembalian `shouldSubscribeToParticipant` tidak diharapkan berubah ketika `STAGE_PARTICIPANT_PUBLISH_STATE_CHANGED` dipanggil. Jika aplikasi host ingin berlangganan ke peserta tertentu, itu harus mengembalikan jenis langganan yang diinginkan terlepas dari status publikasi peserta tersebut. SDK bertanggung jawab untuk memastikan bahwa keadaan strategi yang diinginkan ditindaklanjuti pada waktu yang tepat berdasarkan keadaan tahap.

Publikasikan Aliran Media

Perangkat lokal seperti mikrofon dan kamera diambil menggunakan langkah yang sama seperti yang diuraikan di atas dalam [Ambil MediaStream](#) dari Perangkat. Dalam contoh yang kita gunakan `MediaStream` untuk membuat daftar `LocalStageStream` objek yang digunakan untuk penerbitan oleh SDK:

```
try {
  // Get stream using steps outlined in document above
  const stream = await getMediaStreamFromDevice();

  let streamsToPublish = stream.getTracks().map(track => {
    new LocalStageStream(track)
  });

  // Create stage with strategy, or update existing strategy
  const strategy = {
    stageStreamsToPublish: () => streamsToPublish
  }
}
```

Publikasikan Screenshare

Aplikasi sering perlu mempublikasikan screenshare selain kamera web pengguna. Menerbitkan screenshare mengharuskan membuat tambahan Stage dengan token uniknya sendiri.

```
// Invoke the following lines to get the screenshare's tracks
const media = await navigator.mediaDevices.getDisplayMedia({
  video: {
    width: {
      max: 1280,
    },
    height: {
      max: 720,
    }
  }
});
const screenshare = { videoStream: new LocalStageStream(media.getVideoTracks()[0]) };
const screenshareStrategy = {
  stageStreamsToPublish: () => {
    return [screenshare.videoStream];
  },
  shouldPublishParticipant: (participant) => {
    return true;
  },
  shouldSubscribeToParticipant: (participant) => {
    return SubscribeType.AUDIO_VIDEO;
  }
}
const screenshareStage = new Stage(screenshareToken, screenshareStrategy);
await screenshareStage.join();
```

Tampilkan dan Hapus Peserta

Setelah berlangganan selesai, Anda menerima berbagai StageStream objek melalui `STAGE_PARTICIPANT_STREAMS_ADDED` acara. Acara ini juga memberi Anda info peserta untuk membantu saat menampilkan aliran media:

```
stage.on(StageEvents.STAGE_PARTICIPANT_STREAMS_ADDED, (participant, streams) => {
  const streamsToDisplay = streams;

  if (participant.isLocal) {
    // Ensure to exclude local audio streams, otherwise echo will occur
    streamsToDisplay = streams.filter(stream => stream.streamType ===
StreamType.VIDEO)
  }

  // Create or find video element already available in your application
```

```
const videoEl = getParticipantVideoElement(participant.id);

// Attach the participants streams
videoEl.srcObject = new MediaStream();
streamsToDisplay.forEach(stream =>
  videoEl.srcObject.addTrack(stream.mediaStreamTrack));
})
```

Ketika peserta berhenti menerbitkan atau berhenti berlangganan dari aliran, `STAGE_PARTICIPANT_STREAMS_REMOVED` fungsi dipanggil dengan aliran yang telah dihapus. Aplikasi host harus menggunakan ini sebagai sinyal untuk menghapus aliran video peserta dari DOM.

`STAGE_PARTICIPANT_STREAMS_REMOVED` dipanggil untuk semua skenario di mana aliran mungkin dihapus, termasuk:

- Peserta jarak jauh berhenti menerbitkan.
- Perangkat lokal berhenti berlangganan atau mengubah langganan dari ke `AUDIO_VIDEO`. `AUDIO_ONLY`
- Peserta jarak jauh meninggalkan panggung.
- Peserta lokal meninggalkan panggung.

Karena `STAGE_PARTICIPANT_STREAMS_REMOVED` dipanggil untuk semua skenario, tidak diperlukan logika bisnis khusus untuk menghapus peserta dari UI selama operasi cuti jarak jauh atau lokal.

Bisikan dan Bunyikan Streaming Media

`LocalStageStream` objek memiliki `setMuted` fungsi yang mengontrol apakah aliran diredam. Fungsi ini dapat dipanggil pada aliran sebelum atau sesudah dikembalikan dari fungsi `stageStreamsToPublish` strategi.

Penting: Jika instance `LocalStageStream` objek baru dikembalikan `stageStreamsToPublish` setelah panggilan `refreshStrategy`, status bisu objek aliran baru diterapkan ke panggung. Hati-hati saat membuat `LocalStageStream` instance baru untuk memastikan status bisu yang diharapkan dipertahankan.

Pantau Status Bisu Media Peserta Jarak Jauh

Saat peserta mengubah status bisu video atau audio mereka, `STAGE_STREAM_MUTE_CHANGED` acara dipicu dengan daftar aliran yang telah berubah. Gunakan `isMuted` properti `StageStream` untuk memperbarui UI Anda sesuai:

```
stage.on(StageEvents.STAGE_STREAM_MUTE_CHANGED, (participant, stream) => {
  if (stream.streamType === 'video' && stream.isMuted) {
    // handle UI changes for video track getting muted
  }
})
```

Selain itu, Anda dapat melihat [StageParticipantInfo](#) informasi negara tentang apakah audio atau video diredam:

```
stage.on(StageEvents.STAGE_STREAM_MUTE_CHANGED, (participant, stream) => {
  if (participant.videoStopped || participant.audioMuted) {
    // handle UI changes for either video or audio
  }
})
```

Dapatkan Statistik WebRTC

Untuk mendapatkan statistik WebRTC terbaru untuk aliran penerbitan atau aliran berlangganan, gunakan `getStats` `StageStream`. Ini adalah metode asinkron yang dengannya Anda dapat mengambil statistik baik melalui `await` atau dengan merantai janji. Hasilnya adalah kamus `RTCStatsReport` yang berisi semua statistik standar.

```
try {
  const stats = await stream.getStats();
} catch (error) {
  // Unable to retrieve stats
}
```

Mengoptimalkan Media

Disarankan untuk membatasi `getUserMedia` dan `getDisplayMedia` memanggil ke batasan berikut untuk kinerja terbaik:

```
const CONSTRAINTS = {
  video: {
```



```
    width: { ideal: 1280 }, // Note: flip width and height values if portrait is
desired
    height: { ideal: 720 },
    framerate: { ideal: 30 },
  },
};
```

Anda selanjutnya dapat membatasi media melalui opsi tambahan yang diteruskan ke `LocalStageStream` konstruktor:

```
const localStreamOptions = {
  minBitrate?: number;
  maxBitrate?: number;
  maxFramerate?: number;
  simulcast: {
    enabled: boolean
  }
}
const localStream = new LocalStageStream(track, localStreamOptions)
```

Dalam kode di atas:

- `minBitrate` menetapkan bitrate minimum yang diharapkan digunakan browser. Namun, aliran video dengan kompleksitas rendah dapat mendorong encoder lebih rendah dari bitrate ini.
- `maxBitrate` menetapkan bitrate maksimum yang diharapkan tidak melebihi browser untuk aliran ini.
- `maxFramerate` menetapkan frame rate maksimum yang diharapkan tidak melebihi browser untuk aliran ini.
- `simulcast` Opsi ini hanya dapat digunakan di browser berbasis Chromium. Ini memungkinkan pengiriman tiga lapisan rendisi aliran.
 - Ini memungkinkan server untuk memilih rendisi mana yang akan dikirim ke peserta lain, berdasarkan keterbatasan jaringan mereka.
 - Ketika `simulcast` ditentukan bersama dengan `maxFramerate` nilai `maxBitrate` dan/atau, diharapkan bahwa lapisan rendisi tertinggi akan dikonfigurasi dengan nilai-nilai ini dalam pikiran, asalkan `maxBitrate` tidak berada di bawah `maxBitrate` nilai default lapisan tertinggi kedua SDK internal yaitu 900 kbps.
 - Jika `maxBitrate` ditentukan sebagai terlalu rendah dibandingkan dengan nilai default lapisan tertinggi kedua, `simulcast` akan dinonaktifkan.

- simulcast tidak dapat dinyalakan dan dimatikan tanpa menerbitkan ulang media melalui kombinasi `shouldPublishParticipant` pengembalian `false`, panggilan, pengembalian, `true` dan panggilan `refreshStrategy shouldPublishParticipant` lagi. `refreshStrategy`

Dapatkan Atribut Peserta

Jika Anda menentukan atribut dalam permintaan `CreateParticipantToken` titik akhir, Anda dapat melihat atribut di `StageParticipantInfo` properti:

```
stage.on(StageEvents.STAGE_PARTICIPANT_JOINED, (participant) => {
  console.log(`Participant ${participant.id} info:`, participant.attributes);
})
```

Menangani Masalah Jaringan

Ketika koneksi jaringan perangkat lokal terputus, SDK secara internal mencoba menyambung kembali tanpa tindakan pengguna apa pun. Dalam beberapa kasus, SDK tidak berhasil dan tindakan pengguna diperlukan.

Secara umum keadaan panggung dapat ditangani melalui acara:

`STAGE_CONNECTION_STATE_CHANGED`

```
stage.on(StageEvents.STAGE_CONNECTION_STATE_CHANGED, (state) => {
  switch (state) {
    case StageConnectionState.DISCONNECTED:
      // handle disconnected UI
      break;
    case StageConnectionState.CONNECTING:
      // handle establishing connection UI
      break;
    case StageConnectionState.CONNECTED:
      // SDK is connected to the Stage
      break;
    case StageConnectionState.ERRORRED:
      // unrecoverable error detected, please re-instantiate
      Break;
  }
})
```

Secara umum, menemukan kesalahan setelah berhasil bergabung dengan tahap menunjukkan bahwa SDK kehilangan koneksi dan tidak berhasil membangun kembali koneksi. Buat Stage objek baru dan cobalah untuk bergabung ketika kondisi jaringan membaik.

Siarkan Panggung ke Saluran IVS

Untuk menyiarkan panggung, buat `IVSBroadcastClient` sesi terpisah dan kemudian ikuti instruksi biasa untuk penyiaran dengan SDK, yang dijelaskan di atas. Daftar `StageStream` ekspos via `STAGE_PARTICIPANT_STREAMS_ADDED` dapat digunakan untuk mengambil aliran media peserta yang dapat diterapkan pada komposisi aliran siaran, sebagai berikut:

```
// Setup client with preferred settings
const broadcastClient = getIvsBroadcastClient();

stage.on(StageEvents.STAGE_PARTICIPANT_STREAMS_ADDED, (participant, streams) => {
  streams.forEach(stream => {
    const inputStream = new MediaStream([stream.mediaStreamTrack]);
    switch (stream.streamType) {
      case StreamType.VIDEO:
        broadcastClient.addVideoInputDevice(inputStream, `video-
${participant.id}`, {
          index: DESIRED_LAYER,
          width: MAX_WIDTH,
          height: MAX_HEIGHT
        });
        break;
      case StreamType.AUDIO:
        broadcastClient.addAudioInputDevice(inputStream, `audio-
${participant.id}`);
        break;
    }
  })
})
```

Secara opsional, Anda dapat menggabungkan panggung dan menyiarkannya ke saluran latensi rendah IVS, untuk menjangkau audiens yang lebih besar. Lihat [Mengaktifkan Beberapa Host di Amazon IVS Stream di Panduan Pengguna Streaming Latensi Rendah IVS](#).

Masalah dan Solusi yang Diketahui

- Saat menutup tab browser atau keluar dari browser tanpa menelepon `stage.leave()`, pengguna masih dapat muncul dalam sesi dengan bingkai beku atau layar hitam hingga 10 detik.

Solusi: Tidak ada.

- Sesi Safari sebentar-sebentar muncul dengan layar hitam untuk pengguna yang bergabung setelah sesi dimulai.

Solusi: Segarkan browser dan sambungkan kembali sesi.

- Safari tidak pulih dengan anggun dari switching jaringan.

Solusi: Segarkan browser dan sambungkan kembali sesi.

- Konsol pengembang mengulangi `Error: UnintentionalError at StageSocket.onClose` kesalahan.

Solusi: Hanya satu tahap yang dapat dibuat per token peserta. Kesalahan ini terjadi ketika lebih dari satu Stage instance dibuat dengan token peserta yang sama, terlepas dari apakah instance tersebut ada di satu perangkat atau beberapa perangkat.

- Anda mungkin mengalami kesulitan mempertahankan `StageParticipantPublishState.PUBLISHED` status dan mungkin menerima `StageParticipantPublishState.ATTEMPTING_PUBLISH` status berulang saat mendengarkan `StageEvents.STAGE_PARTICIPANT_PUBLISH_STATE_CHANGED` acara tersebut.

Solusi: Batasi resolusi video ke 720p saat memanggil atau `getUserMedia` `getDisplayMedia`. Secara khusus, nilai `getUserMedia` dan `getDisplayMedia` batasan untuk lebar dan tinggi tidak boleh melebihi 921600 (1280* 720) saat dikalikan bersama.

Keterbatasan Safari

- Menolak prompt izin memerlukan pengaturan ulang izin di pengaturan situs web Safari di tingkat OS.
- Safari tidak secara native mendeteksi semua perangkat seefektif Firefox atau Chrome. Misalnya, Kamera Virtual OBS tidak terdeteksi.

Keterbatasan Firefox

- Izin sistem harus diaktifkan agar Firefox dapat berbagi layar. Setelah mengaktifkannya, pengguna harus me-restart Firefox agar berfungsi dengan benar; jika tidak, jika izin dianggap diblokir, browser akan memberikan [NotFoundError](#) pengecualian.
- `getCapabilities` metodenya hilang. Ini berarti pengguna tidak bisa mendapatkan resolusi atau rasio aspek trek media. Lihat utas [bugzilla](#) ini.
- Beberapa `AudioContext` properti hilang; misalnya, latensi dan jumlah saluran. Ini bisa menimbulkan masalah bagi pengguna tingkat lanjut yang ingin memanipulasi trek audio.
- Umpan kamera `getUserMedia` dibatasi hingga rasio aspek 4:3 di macOS. Lihat utas [bugzilla 1](#) dan utas [bugzilla 2](#).
- Pengambilan audio tidak didukung dengan `getDisplayMedia`. Lihat utas [bugzilla](#) ini.
- `Framerate` dalam tangkapan layar kurang optimal (sekitar 15fps?). Lihat utas [bugzilla](#) ini.

Batasan Web Seluler

- [getDisplayMedia](#) berbagi layar tidak didukung di perangkat seluler.

Solusi: Tidak ada.

- Peserta membutuhkan waktu 15-30 detik untuk pergi saat menutup browser tanpa menelepon `leave()`.

Solusi: Tambahkan UI yang mendorong pengguna untuk memutuskan sambungan dengan benar.

- Aplikasi latar belakang menyebabkan penerbitan video berhenti.

Solusi: Menampilkan papan tulis UI saat penerbit diijeda.

- Video framerate turun selama kurang lebih 5 detik setelah mematikan kamera di perangkat Android.

Solusi: Tidak ada.

- Umpan video diregangkan pada rotasi untuk iOS 16.0.

Solusi: Tampilkan UI yang menguraikan masalah OS yang diketahui ini.

- Mengalihkan perangkat input audio secara otomatis mengalihkan perangkat output audio.

Solusi: Tidak ada.

- Latar belakang browser menyebabkan aliran penerbitan menjadi hitam dan hanya menghasilkan audio.

Solusi: Tidak ada. Ini untuk alasan keamanan.

Penanganan Kesalahan

Bagian ini adalah ikhtisar kondisi kesalahan, bagaimana Web Broadcast SDK melaporkannya ke aplikasi, dan apa yang harus dilakukan aplikasi ketika kesalahan tersebut ditemui. Ada empat kategori kesalahan:

```
try {
  stage = new Stage(token, strategy);
} catch (e) {
  // 1) stage instantiation errors
}

try {
  await stage.join();
} catch (e) {
  // 2) stage join errors
}

stage.on(StageEvents.STAGE_PARTICIPANT_PUBLISH_STATE_CHANGED, (participantInfo, state)
=> {
  if (state === StageParticipantPublishState.ERRORRED) {
    // 3) stage publish errors
  }
});

stage.on(StageEvents.STAGE_PARTICIPANT_SUBSCRIBE_STATE_CHANGED, (participantInfo,
state) => {
  if (state === StageParticipantSubscribeState.ERRORRED) {
    // 4) stage subscribe errors
  }
});
```

Kesalahan Instantiasi Tahap

Instantiasi tahap tidak memvalidasi token dari jarak jauh, tetapi memeriksa beberapa masalah token dasar yang dapat divalidasi di sisi klien. Akibatnya, SDK dapat menimbulkan kesalahan.

Token Peserta Cacat

Ini terjadi ketika token panggung salah bentuk. Saat membuat instance Stage, SDK memunculkan error dengan pesan ini: "Error parsing Stage Token."

Tindakan: Buat token yang valid dan coba lagi membuat instance.

Kesalahan Gabung Panggung

Ini adalah kesalahan yang mungkin terjadi ketika awalnya mencoba untuk bergabung dengan panggung.

Panggung telah dihapus

Ini terjadi ketika bergabung dengan tahap (terkait dengan token) yang dihapus. Metode `join` SDK melempar kesalahan dengan pesan ini: "InitialConnectTimedOut setelah 10 detik."

Tindakan: Buat token yang valid dengan tahap baru dan coba lagi bergabung.

Token Peserta Kedaluwarsa

Ini terjadi ketika token kedaluwarsa. Metode `join` SDK memunculkan kesalahan dengan pesan ini: "Token kedaluwarsa dan tidak lagi valid."

Tindakan: Buat token baru dan coba lagi bergabung.

Token Peserta Tidak Valid atau Dicabut

Ini terjadi ketika token tidak valid atau dicabut/terputus. Metode `join` SDK melempar kesalahan dengan pesan ini: "InitialConnectTimedOut setelah 10 detik."

Tindakan: Buat token baru dan coba lagi bergabung.

Token Terputus

Ini terjadi ketika token panggung tidak cacat tetapi ditolak oleh server Stages. Metode `join` SDK melempar kesalahan dengan pesan ini: "InitialConnectTimedOut setelah 10 detik."

Tindakan: Buat token yang valid dan coba lagi bergabung.

Kesalahan Jaringan untuk Gabung Awal

Ini terjadi ketika SDK tidak dapat menghubungi server Stages untuk membuat koneksi. Metode `join` SDK melempar kesalahan dengan pesan ini: "InitialConnectTimedOut setelah 10 detik."

Tindakan: Tunggu konektivitas perangkat pulih dan coba lagi bergabung.

Kesalahan Jaringan saat Sudah Bergabung

Jika koneksi jaringan perangkat mati, SDK mungkin kehilangan koneksinya ke server Stage. Anda mungkin melihat kesalahan di konsol karena SDK tidak dapat lagi menjangkau layanan backend. Posting ke <https://broadcast.stats.live-video.net> akan gagal.

Jika Anda menerbitkan dan/atau berlangganan, Anda akan melihat kesalahan di konsol terkait dengan upaya untuk menerbitkan/berlangganan.

Secara internal SDK akan mencoba terhubung kembali dengan strategi backoff eksponensial.

Tindakan: Tunggu konektivitas perangkat pulih. Jika menerbitkan atau berlangganan, segarkan strategi untuk memastikan publikasi ulang aliran media Anda.

Publikasikan dan Berlangganan Kesalahan

Publikasikan Kesalahan: Publikasikan Status

SDK melaporkan `ERRORRED` saat publikasi gagal. Hal ini dapat terjadi karena kondisi jaringan atau jika suatu tahap berada pada kapasitas untuk penerbit.

```
stage.on(StageEvents.STAGE_PARTICIPANT_PUBLISH_STATE_CHANGED, (participantInfo, state)
=> {
  if (state === StageParticipantPublishState.ERRORRED) {
    // Handle
  }
});
```

Tindakan: Segarkan strategi untuk mencoba publikasi ulang aliran media Anda.

Kesalahan Berlangganan

SDK melaporkan `ERRORRED` saat berlangganan gagal. Ini dapat terjadi karena kondisi jaringan atau jika suatu tahap berada pada kapasitas untuk pelanggan.


```
stage.on(StageEvents.STAGE_PARTICIPANT_SUBSCRIBE_STATE_CHANGED, (participantInfo,
state) => {
  if (state === StageParticipantSubscribeState.ERRORRED) {
    // 4) stage subscribe errors
  }
});
```

Tindakan: Segarkan strategi untuk mencoba berlangganan baru.

SDK Siaran IVS: Panduan Android (Streaming Waktu Nyata)

IVS real-time streaming Android broadcast SDK memungkinkan peserta untuk mengirim dan menerima video di Android.

`com.amazonaws.ivs.broadcast` Paket mengimplementasikan antarmuka yang dijelaskan dalam dokumen ini. SDK mendukung operasi berikut:

- Bergabunglah dengan panggung
- Publikasikan media ke peserta lain di panggung
- Berlangganan media dari peserta lain di panggung
- Kelola dan pantau video dan audio yang dipublikasikan ke panggung
- Dapatkan statistik WebRTC untuk setiap koneksi rekan
- Semua operasi dari SDK siaran Android streaming latensi rendah IVS

Versi terbaru SDK siaran Android: [1.14.1 \(Catatan Rilis\)](#)

Dokumentasi referensi: Untuk informasi tentang metode terpenting yang tersedia di SDK siaran Android Amazon IVS, lihat dokumentasi referensi di <https://aws.github.io/amazon-ivs-broadcast-docs/1.14.1/android/>.

Kode contoh: Lihat repositori contoh Android di GitHub: [https://github.com/aws-samples/amazon-ivs-broadcast-android](https://github.com/aws-samples/amazon-ivs-broadcast-android-sample) -sample.

Persyaratan platform: Android 9.0 dan yang lebih baru.

Memulai

Instal Perpustakaan

Untuk menambahkan library siaran Amazon IVS Android ke lingkungan pengembangan Android Anda, tambahkan pustaka ke `build.gradle` file modul Anda, seperti yang ditunjukkan di sini (untuk versi terbaru SDK siaran Amazon IVS):

```
repositories {
    mavenCentral()
}

dependencies {
    implementation 'com.amazonaws:ivs-broadcast:1.14.1:stages@aar'
}
```

Tambahkan izin berikut ke manifes Anda untuk mengizinkan SDK mengaktifkan dan menonaktifkan speakerphone:

```
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS"/>
```

Sebagai alternatif, untuk menginstal SDK secara manual, unduh versi terbaru dari lokasi ini:

<https://search.maven.org/artifact/com.amazonaws/ivs-broadcast>

Pastikan untuk mengunduh `aar` dengan `-stages` menambahkan.

Permintaan Izin

Aplikasi Anda harus meminta izin untuk mengakses kamera dan mikrofon pengguna. (Ini tidak spesifik untuk Amazon IVS; diperlukan untuk aplikasi apa pun yang membutuhkan akses ke kamera dan mikrofon.)

Di sini, kami memeriksa apakah pengguna telah memberikan izin dan, jika tidak, memintanya:

```
final String[] requiredPermissions =
    { Manifest.permission.CAMERA, Manifest.permission.RECORD_AUDIO };

for (String permission : requiredPermissions) {
    if (ContextCompat.checkSelfPermission(this, permission)
        != PackageManager.PERMISSION_GRANTED) {
```

```
// If any permissions are missing we want to just request them all.
ActivityCompat.requestPermissions(this, requiredPermissions, 0x100);
break;
}
}
```

Di sini, kami mendapatkan respons pengguna:

```
@Override
public void onRequestPermissionsResult(int requestCode,
                                     @NonNull String[] permissions,
                                     @NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode,
                                    permissions, grantResults);
    if (requestCode == 0x100) {
        for (int result : grantResults) {
            if (result == PackageManager.PERMISSION_DENIED) {
                return;
            }
        }
        setupBroadcastSession();
    }
}
```

Penerbitan dan Berlangganan

Konsep

Tiga konsep inti mendasari fungsionalitas real-time: [panggung](#), [strategi](#), dan [penyaji](#). Tujuan desain adalah meminimalkan jumlah logika sisi klien yang diperlukan untuk membangun produk yang berfungsi.

Tahap

StageKelas adalah titik utama interaksi antara aplikasi host dan SDK. Ini mewakili panggung itu sendiri dan digunakan untuk bergabung dan meninggalkan panggung. Membuat dan menggabungkan tahap membutuhkan string token yang valid dan belum kedaluwarsa dari bidang kontrol (direpresentasikan sebagai token). Bergabung dan meninggalkan panggung itu sederhana.

```
Stage stage = new Stage(context, token, strategy);

try {
```

```
stage.join();
} catch (BroadcastException exception) {
    // handle join exception
}

stage.leave();
```

StageKelas juga di mana StageRendererer dapat dilampirkan:

```
stage.addRenderer(renderer); // multiple renderers can be added
```

Strategi

Stage.StrategyAntarmuka menyediakan cara bagi aplikasi host untuk mengkomunikasikan status tahap yang diinginkan ke SDK. Tiga fungsi perlu diimplementasikan: `shouldSubscribeToParticipant`, `shouldPublishFromParticipant`, dan `stageStreamsToPublishForParticipant`. Semua dibahas di bawah ini.

Berlangganan Peserta

```
Stage.SubscribeType shouldSubscribeToParticipant(@NonNull Stage stage, @NonNull
ParticipantInfo participantInfo);
```

Ketika peserta jarak jauh bergabung dengan panggung, SDK menanyakan aplikasi host tentang status langganan yang diinginkan untuk peserta tersebut. Pilihannya adalah `NONE`, `AUDIO_ONLY`, dan `AUDIO_VIDEO`. Saat mengembalikan nilai untuk fungsi ini, aplikasi host tidak perlu khawatir tentang status publikasi, status langganan saat ini, atau status koneksi tahap. Jika `AUDIO_VIDEO` dikembalikan, SDK menunggu hingga peserta jarak jauh memublikasikan sebelum berlangganan, dan memperbarui aplikasi host melalui perender selama proses berlangsung.

Berikut adalah contoh implementasi:

```
@Override
Stage.SubscribeType shouldSubscribeToParticipant(@NonNull Stage stage, @NonNull
ParticipantInfo participantInfo) {
    return Stage.SubscribeType.AUDIO_VIDEO;
}
```

Ini adalah implementasi lengkap dari fungsi ini untuk aplikasi host yang selalu ingin semua peserta untuk melihat satu sama lain; misalnya, aplikasi obrolan video.

Implementasi yang lebih maju juga dimungkinkan. Gunakan `userInfo` properti `ParticipantInfo` untuk berlangganan peserta secara selektif berdasarkan atribut yang disediakan server:

```
@Override
Stage.SubscribeType shouldSubscribeToParticipant(@NonNull Stage stage, @NonNull
ParticipantInfo participantInfo) {
    switch(participantInfo.userInfo.get("role")) {
        case "moderator":
            return Stage.SubscribeType.NONE;
        case "guest":
            return Stage.SubscribeType.AUDIO_VIDEO;
        default:
            return Stage.SubscribeType.NONE;
    }
}
```

Ini dapat digunakan untuk membuat panggung di mana moderator dapat memantau semua tamu tanpa terlihat atau didengar sendiri. Aplikasi host dapat menggunakan logika bisnis tambahan untuk membiarkan moderat melihat satu sama lain tetapi tetap tidak terlihat oleh tamu.

Penerbitan

```
boolean shouldPublishFromParticipant(@NonNull Stage stage, @NonNull ParticipantInfo
participantInfo);
```

Setelah terhubung ke panggung, SDK menanyakan aplikasi host untuk melihat apakah peserta tertentu harus mempublikasikannya. Ini hanya dipanggil pada peserta lokal yang memiliki izin untuk mempublikasikan berdasarkan token yang disediakan.

Berikut adalah contoh implementasi:

```
@Override
boolean shouldPublishFromParticipant(@NonNull Stage stage, @NonNull ParticipantInfo
participantInfo) {
    return true;
}
```

Ini untuk aplikasi obrolan video standar di mana pengguna selalu ingin mempublikasikan. Mereka dapat membisukan dan menonaktifkan audio dan video mereka, untuk langsung disembunyikan atau dilihat/didengar. (Mereka juga dapat menggunakan `publish/unpublish`, tetapi itu jauh lebih lambat. `Mute/unmute` lebih disukai untuk kasus penggunaan di mana perubahan visibilitas sering diinginkan.)

Memilih Streaming untuk Publikasikan

```
@Override
List<LocalStageStream> stageStreamsToPublishForParticipant(@NonNull Stage stage,
    @NonNull ParticipantInfo participantInfo);
}
```

Saat menerbitkan, ini digunakan untuk menentukan aliran audio dan video apa yang harus dipublikasikan. Ini dibahas secara lebih rinci nanti di [Publish a Media Stream](#).

Memperbarui Strategi

Strategi ini dimaksudkan untuk menjadi dinamis: nilai yang dikembalikan dari salah satu fungsi di atas dapat diubah kapan saja. Misalnya, jika aplikasi host tidak ingin mempublikasikan sampai pengguna akhir menyetuk tombol, Anda dapat mengembalikan variabel dari `shouldPublishFromParticipant` (sesuatu seperti `hasUserTappedPublishButton`). Ketika variabel itu berubah berdasarkan interaksi oleh pengguna akhir, panggil `stage.refreshStrategy()` untuk memberi sinyal ke SDK bahwa ia harus menanyakan strategi untuk nilai terbaru, hanya menerapkan hal-hal yang telah berubah. Jika SDK mengamati bahwa `shouldPublishFromParticipant` nilainya telah berubah, SDK akan memulai proses publikasi. Jika kueri SDK dan semua fungsi mengembalikan nilai yang sama seperti sebelumnya, `refreshStrategy` panggilan tidak akan melakukan modifikasi apa pun pada tahapan.

Jika nilai pengembalian `shouldSubscribeToParticipant` perubahan dari `AUDIO_VIDEO` ke `AUDIO_ONLY`, aliran video akan dihapus untuk semua peserta dengan nilai yang dikembalikan diubah, jika aliran video ada sebelumnya.

Umumnya, tahap menggunakan strategi untuk menerapkan perbedaan antara strategi sebelumnya dan saat ini secara efisien, tanpa aplikasi host perlu khawatir tentang semua keadaan yang diperlukan untuk mengelolanya dengan benar. Karena itu, anggap menelepon `stage.refreshStrategy()` sebagai operasi yang murah, karena tidak melakukan apa-apa kecuali strateginya berubah.

Penyaji

`StageRendererAntarmuka` mengkomunikasikan keadaan panggung ke aplikasi host. Pembaruan pada UI aplikasi host biasanya dapat didukung sepenuhnya oleh peristiwa yang disediakan oleh perender. Penyaji menyediakan fungsi-fungsi berikut:

```
void onParticipantJoined(@NonNull Stage stage, @NonNull ParticipantInfo
    participantInfo);

void onParticipantLeft(@NonNull Stage stage, @NonNull ParticipantInfo participantInfo);

void onParticipantPublishStateChanged(@NonNull Stage stage, @NonNull ParticipantInfo
    participantInfo, @NonNull Stage.PublishState publishState);

void onParticipantSubscribeStateChanged(@NonNull Stage stage, @NonNull ParticipantInfo
    participantInfo, @NonNull Stage.SubscribeState subscribeState);

void onStreamsAdded(@NonNull Stage stage, @NonNull ParticipantInfo participantInfo,
    @NonNull List<StageStream> streams);

void onStreamsRemoved(@NonNull Stage stage, @NonNull ParticipantInfo participantInfo,
    @NonNull List<StageStream> streams);

void onStreamsMutedChanged(@NonNull Stage stage, @NonNull ParticipantInfo
    participantInfo, @NonNull List<StageStream> streams);

void onError(@NonNull BroadcastException exception);

void onConnectionStateChanged(@NonNull Stage stage, @NonNull Stage.ConnectionState
    state, @Nullable BroadcastException exception);
```

Untuk sebagian besar metode ini, yang sesuai Stage dan ParticipantInfo disediakan.

Tidak diharapkan bahwa informasi yang diberikan oleh penyaji berdampak pada nilai pengembalian strategi. Misalnya, nilai pengembalian `shouldSubscribeToParticipant` tidak diharapkan berubah ketika `onParticipantPublishStateChanged` dipanggil. Jika aplikasi host ingin berlangganan ke peserta tertentu, itu harus mengembalikan jenis langganan yang diinginkan terlepas dari status publikasi peserta tersebut. SDK bertanggung jawab untuk memastikan bahwa keadaan strategi yang diinginkan ditindaklanjuti pada waktu yang tepat berdasarkan keadaan tahap.

StageRenderer dapat dilampirkan ke kelas panggung:

```
stage.addRenderer(renderer); // multiple renderers can be added
```

Perhatikan bahwa hanya peserta penerbitan yang dipicu `onParticipantJoined`, dan setiap kali peserta berhenti menerbitkan atau meninggalkan sesi panggung, `onParticipantLeft` dipicu.

Publikasikan Aliran Media

Perangkat lokal seperti mikrofon dan kamera internal ditemukan melalui `DeviceDiscovery`. Berikut adalah contoh memilih kamera yang menghadap ke depan dan mikrofon default, lalu mengembalikannya seperti `LocalStageStreams` yang akan dipublikasikan oleh SDK:

```
DeviceDiscovery deviceDiscovery = new DeviceDiscovery(context);

List<Device> devices = deviceDiscovery.listLocalDevices();
List<LocalStageStream> publishStreams = new ArrayList<LocalStageStream>();

Device frontCamera = null;
Device microphone = null;

// Create streams using the front camera, first microphone
for (Device device : devices) {
    Device.Descriptor descriptor = device.getDescriptor();
    if (!frontCamera && descriptor.type == Device.Descriptor.DeviceType.Camera &&
        descriptor.position == Device.Descriptor.Position.FRONT) {
        frontCamera = device;
    }
    if (!microphone && descriptor.type == Device.Descriptor.DeviceType.Microphone) {
        microphone = device;
    }
}

ImageLocalStageStream cameraStream = new ImageLocalStageStream(frontCamera);
AudioLocalStageStream microphoneStream = new AudioLocalStageStream(microphoneDevice);

publishStreams.add(cameraStream);
publishStreams.add(microphoneStream);

// Provide the streams in Stage.Strategy
@Override
@NonNull List<LocalStageStream> stageStreamsToPublishForParticipant(@NonNull Stage
    stage, @NonNull ParticipantInfo participantInfo) {
    return publishStreams;
}
```


Tampilkan dan Hapus Peserta

Setelah berlangganan selesai, Anda akan menerima array `StageStream` objek melalui fungsi `renderer.onStreamsAdded` Anda dapat mengambil pratinjau dari: `ImageStageStream`

```
ImagePreviewView preview = ((ImageStageStream)stream).getPreview();

// Add the view to your view hierarchy
LinearLayout previewHolder = findViewById(R.id.previewHolder);
preview.setLayoutParams(new LinearLayout.LayoutParams(
    LinearLayout.LayoutParams.MATCH_PARENT,
    LinearLayout.LayoutParams.MATCH_PARENT));
previewHolder.addView(preview);
```

Anda dapat mengambil statistik tingkat audio dari: `AudioStageStream`

```
((AudioStageStream)stream).setStatsCallback((peak, rms) -> {
    // handle statistics
});
```

Ketika peserta berhenti menerbitkan atau berhenti berlangganan, `onStreamsRemoved` fungsi dipanggil dengan aliran yang telah dihapus. Aplikasi host harus menggunakan ini sebagai sinyal untuk menghapus aliran video peserta dari hierarki tampilan.

`onStreamsRemoved` dipanggil untuk semua skenario di mana aliran mungkin dihapus, termasuk:

- Peserta jarak jauh berhenti menerbitkan.
- Perangkat lokal berhenti berlangganan atau mengubah langganan dari ke `AUDIO_VIDEO`.
`AUDIO_ONLY`
- Peserta jarak jauh meninggalkan panggung.
- Peserta lokal meninggalkan panggung.

Karena `onStreamsRemoved` dipanggil untuk semua skenario, tidak diperlukan logika bisnis khusus untuk menghapus peserta dari UI selama operasi cuti jarak jauh atau lokal.

Bisikan dan Bunyikan Streaming Media

`LocalStageStream` objek memiliki `setMuted` fungsi yang mengontrol apakah aliran diredam. Fungsi ini dapat dipanggil pada aliran sebelum atau sesudah dikembalikan dari fungsi `streamsToPublishForParticipant` strategi.

Penting: Jika instance `LocalStageStream` objek baru dikembalikan `streamsToPublishForParticipant` setelah panggilan `refreshStrategy`, status bisu objek aliran baru diterapkan ke panggung. Hati-hati saat membuat `LocalStageStream` instance baru untuk memastikan status bisu yang diharapkan dipertahankan.

Pantau Status Bisu Media Peserta Jarak Jauh

Saat peserta mengubah status bisu aliran video atau audio mereka, `onStreamMutedChanged` fungsi penyaji dipanggil dengan daftar aliran yang telah berubah. Gunakan `getMuted` metode ini `StageStream` untuk memperbarui UI Anda sesuai dengan itu.

```
@Override
void onStreamsMutedChanged(@NonNull Stage stage, @NonNull ParticipantInfo
    participantInfo, @NonNull List<StageStream> streams) {
    for (StageStream stream : streams) {
        boolean muted = stream.getMuted();
        // handle UI changes
    }
}
```

Dapatkan Statistik WebRTC

Untuk mendapatkan statistik WebRTC terbaru untuk aliran penerbitan atau aliran berlangganan, gunakan terus `requestRTCStats` `StageStream` Ketika koleksi selesai, Anda akan menerima statistik melalui `StageStream.Listener` yang dapat diatur `StageStream`.

```
stream.requestRTCStats();

@Override
void onRTCStats(Map<String, Map<String, String>> statsMap) {
    for (Map.Entry<String, Map<String, String>> stat : statsMap.entrySet()) {
        for (Map.Entry<String, String> member : stat.getValue().entrySet()) {
            Log.i(TAG, stat.getKey() + " has member " + member.getKey() + " with value " +
                member.getValue());
        }
    }
}
```

```

    }
  }
}

```

Dapatkan Atribut Peserta

Jika Anda menentukan atribut dalam permintaan `CreateParticipantToken` titik akhir, Anda dapat melihat atribut di `ParticipantInfo` properti:

```

@Override
void onParticipantJoined(@NonNull Stage stage, @NonNull ParticipantInfo
    participantInfo) {
    for (Map.Entry<String, String> entry : participantInfo.userInfo.entrySet()) {
        Log.i(TAG, "attribute: " + entry.getKey() + " = " + entry.getValue());
    }
}

```

Lanjutkan Sesi di Latar Belakang

Saat aplikasi memasuki latar belakang, Anda mungkin ingin berhenti menerbitkan atau berlangganan hanya audio peserta jarak jauh lainnya. Untuk mencapai hal ini, perbarui `Strategy` implementasi Anda untuk menghentikan penerbitan, dan berlangganan `AUDIO_ONLY` (atau `NONE`, jika ada).

```

// Local variables before going into the background
boolean shouldPublish = true;
Stage.SubscribeType subscribeType = Stage.SubscribeType.AUDIO_VIDEO;

// Stage.Strategy implementation
@Override
boolean shouldPublishFromParticipant(@NonNull Stage stage, @NonNull ParticipantInfo
    participantInfo) {
    return shouldPublish;
}

@Override
Stage.SubscribeType shouldSubscribeToParticipant(@NonNull Stage stage, @NonNull
    ParticipantInfo participantInfo) {
    return subscribeType;
}

// In our Activity, modify desired publish/subscribe when we go to background, then
    call refreshStrategy to update the stage

```

```
@Override
void onStop() {
    super.onStop();
    shouldPublish = false;
    subscribeType = Stage.SubscribeType.AUDIO_ONLY;
    stage.refreshStrategy();
}
```

Aktifkan/Nonaktifkan Layered Encoding dengan Simulcast

Saat menerbitkan aliran media, SDK mentransmisikan streaming video berkualitas tinggi dan berkualitas rendah, sehingga peserta jarak jauh dapat berlangganan streaming meskipun mereka memiliki bandwidth downlink terbatas. Pengkodean berlapis dengan simulcast aktif secara default. Anda dapat menonaktifkannya dengan menggunakan `StageVideoConfiguration.Simulcast` kelas:

```
// Disable Simulcast
StageVideoConfiguration config = new StageVideoConfiguration();
config.simulcast.setEnabled(false);

ImageLocalStageStream cameraStream = new ImageLocalStageStream(frontCamera, config);

// Other Stage implementation code
```

Batasan Konfigurasi Video

SDK tidak mendukung pemaksaan mode potret atau mode lanskap menggunakan `StageVideoConfiguration.setSize(BroadcastConfiguration.Vec2 size)`. Dalam orientasi potret, dimensi yang lebih kecil digunakan sebagai lebar; dalam orientasi lanskap, tinggi. Ini berarti bahwa dua panggilan berikut `setSize` memiliki efek yang sama pada konfigurasi video:

```
StageVideo Configuration config = new StageVideo Configuration();

config.setSize(BroadcastConfiguration.Vec2(720f, 1280f);
config.setSize(BroadcastConfiguration.Vec2(1280f, 720f);
```

Menangani Masalah Jaringan

Ketika koneksi jaringan perangkat lokal terputus, SDK secara internal mencoba menyambung kembali tanpa tindakan pengguna apa pun. Dalam beberapa kasus, SDK tidak berhasil dan tindakan pengguna diperlukan. Ada dua kesalahan utama yang terkait dengan kehilangan koneksi jaringan:

- Kode kesalahan 1400, pesan: "PeerConnection hilang karena kesalahan jaringan yang tidak diketahui"
- Kode kesalahan 1300, pesan: "Coba lagi upaya habis"

Jika kesalahan pertama diterima tetapi yang kedua tidak, SDK masih terhubung ke panggung dan akan mencoba membangun kembali koneksinya secara otomatis. Sebagai pengaman, Anda dapat memanggil `refreshStrategy` tanpa perubahan apa pun pada nilai pengembalian metode strategi, untuk memicu upaya penyambungan kembali manual.

Jika kesalahan kedua diterima, upaya penyambungan kembali SDK gagal dan perangkat lokal tidak lagi terhubung ke panggung. Dalam hal ini, cobalah untuk bergabung kembali dengan panggung dengan menelepon `join` setelah koneksi jaringan Anda dibangun kembali.

Secara umum, menemukan kesalahan setelah bergabung dengan tahap berhasil menunjukkan bahwa SDK tidak berhasil membangun kembali koneksi. Buat Stage objek baru dan cobalah untuk bergabung ketika kondisi jaringan membaik.

Menggunakan Mikrofon Bluetooth

Untuk mempublikasikan menggunakan perangkat mikrofon Bluetooth, Anda harus memulai koneksi Bluetooth SCO:

```
Bluetooth.startBluetoothSco(context);  
// Now bluetooth microphones can be used  
...  
// Must also stop bluetooth SCO  
Bluetooth.stopBluetoothSco(context);
```

Masalah dan Solusi yang Diketahui

- Saat perangkat Android tertidur dan bangun, pratinjau mungkin dalam keadaan beku.

Solusi: Buat dan gunakan yang baru. Stage

- Ketika peserta bergabung dengan token yang digunakan oleh peserta lain, koneksi pertama terputus tanpa kesalahan tertentu.

Solusi: Tidak ada.

- Ada masalah langka di mana penerbit menerbitkan tetapi status publikasi yang diterima pelanggan adalah `inactive`.

Solusi: Coba pergi dan kemudian bergabung dengan sesi. Jika masalah tetap ada, buat token baru untuk penerbit.

- Masalah distorsi audio yang jarang terjadi dapat terjadi sebentar-sebentar selama sesi panggung, biasanya pada panggilan dengan durasi yang lebih lama.

Solusi: Peserta dengan audio yang terdistorsi dapat meninggalkan dan bergabung kembali dengan sesi, atau membatalkan publikasi dan menerbitkan ulang audio mereka untuk memperbaiki masalah.

- Mikrofon eksternal tidak didukung saat menerbitkan ke panggung.

Solusi: Jangan gunakan mikrofon eksternal yang terhubung melalui USB untuk menerbitkan ke panggung.

- Penerbitan ke panggung dengan berbagi layar menggunakan `createSystemCaptureSources` tidak didukung.

Solusi: Kelola pengambilan sistem secara manual, menggunakan sumber input gambar khusus dan sumber input audio khusus.

- Ketika `ImagePreviewView` dihapus dari induk (misalnya, `removeView()` dipanggil di induk), segera `ImagePreviewView` dilepaskan. `ImagePreviewView` tidak menampilkan bingkai apa pun saat ditambahkan ke tampilan induk lain.

Solusi: Minta pratinjau lain menggunakan `getPreview`

- Saat bergabung dengan panggung dengan Samsung Galaxy S22/+ dengan Android 12, Anda mungkin mengalami kesalahan 1401 dan perangkat lokal gagal bergabung dengan panggung atau bergabung tetapi tidak memiliki audio.

Solusi: Tingkatkan ke Android 13.

- Saat bergabung dengan panggung dengan Nokia X20 di Android 13, kamera mungkin gagal dibuka dan pengecualian dilemparkan.

Solusi: Tidak ada.

- Perangkat dengan chipset MediaTek Helio mungkin tidak merender video peserta jarak jauh dengan benar.

Solusi: Tidak ada.

- Pada beberapa perangkat, OS perangkat dapat memilih mikrofon yang berbeda dari yang dipilih melalui SDK. Ini karena Amazon IVS Broadcast SDK tidak dapat mengontrol bagaimana rute `VOICE_COMMUNICATION` audio didefinisikan, karena bervariasi sesuai dengan produsen perangkat yang berbeda.

Solusi: Tidak ada.

- Beberapa encoder video Android tidak dapat dikonfigurasi dengan ukuran video kurang dari 176x176. Mengkonfigurasi ukuran yang lebih kecil menyebabkan kesalahan dan mencegah streaming.

Solusi: Jangan mengonfigurasi ukuran video menjadi kurang dari 176x176.

Penanganan Kesalahan

Kesalahan Fatal vs Non-Fatal

Objek kesalahan memiliki bidang boolean “fatal” dari `BroadCastException`

Secara umum, kesalahan fatal terkait dengan koneksi ke server Tahapan (baik koneksi tidak dapat dibuat atau hilang dan tidak dapat dipulihkan). Aplikasi harus membuat ulang panggung dan bergabung kembali, mungkin dengan token baru atau ketika konektivitas perangkat pulih.

Kesalahan non-fatal umumnya terkait dengan status terbitkan/berlangganan dan ditangani oleh SDK, yang mencoba kembali operasi terbitkan/berlangganan.

Anda dapat memeriksa properti ini:

```
try {
    stage.join(...)
} catch (e: BroadCastException) {
    If (e.isFatal) {
        // the error is fatal
    }
}
```

Bergabung Error

Token Cacat

Ini terjadi ketika token panggung salah bentuk.

SDK melempar pengecualian Java dari panggilan `kestage.join`, dengan kode kesalahan = 1000 dan `fatal = true`.

Tindakan: Buat token yang valid dan coba lagi bergabung.

Token Kadaluwarsa

Ini terjadi ketika token panggung kedaluwarsa.

SDK melempar pengecualian Java dari panggilan `kestage.join`, dengan kode kesalahan = 1001 dan `fatal = true`.

Tindakan: Buat token baru dan coba lagi bergabung.

Token Tidak Valid atau Dicabut

Ini terjadi ketika token panggung tidak cacat tetapi ditolak oleh server Stages. Ini dilaporkan secara asinkron melalui perender tahap yang disediakan aplikasi.

SDK memanggil `onConnectionStateChanged` dengan pengecualian, dengan kode kesalahan = 1026 dan `fatal = true`.

Tindakan: Buat token yang valid dan coba lagi bergabung.

Kesalahan Jaringan untuk Gabung Awal

Ini terjadi ketika SDK tidak dapat menghubungi server Stages untuk membuat koneksi. Ini dilaporkan secara asinkron melalui perender tahap yang disediakan aplikasi.

SDK memanggil `onConnectionStateChanged` dengan pengecualian, dengan kode kesalahan = 1300 dan `fatal = true`.

Tindakan: Tunggu konektivitas perangkat pulih dan coba lagi bergabung.

Kesalahan Jaringan saat Sudah Bergabung

Jika koneksi jaringan perangkat mati, SDK mungkin kehilangan koneksinya ke server Stage. Ini dilaporkan secara asinkron melalui perender tahap yang disediakan aplikasi.

SDK memanggil `onConnectionStateChanged` dengan pengecualian, dengan kode kesalahan = 1300 dan `fatal = true`.

Tindakan: Tunggu konektivitas perangkat pulih dan coba lagi bergabung.

Kesalahan Publikasi/Berlangganan

Awal

Ada beberapa kesalahan:

- `MultihostSessionOfferCreationFailPublish (1020)`
- `MultihostSessionOfferCreationFailSubscribe (1021)`
- `MultihostSessionNolceCandidates (1022)`
- `MultihostSessionStageAtCapacity (1024)`
- `SignallingSessionCannotRead (1201)`
- `SignallingSessionCannotSend (1202)`
- `SignallingSessionBadResponse (1203)`

Ini dilaporkan secara asinkron melalui perender tahap yang disediakan aplikasi.

SDK mencoba ulang operasi untuk beberapa kali. Selama percobaan ulang, status terbitkan/berlangganan adalah/. `ATTEMPTING_PUBLISH` `ATTEMPTING_SUBSCRIBE` Jika upaya coba lagi berhasil, status berubah menjadi `PUBLISHED`/`SUBSCRIBED`.

Panggilan SDK `onError` dengan kode kesalahan yang relevan dan `fatal = false`.

Tindakan: Tidak diperlukan tindakan, karena SDK mencoba ulang secara otomatis. Secara opsional, aplikasi dapat menyegarkan strategi untuk memaksa lebih banyak percobaan ulang.

Sudah Didirikan, Lalu Gagal

Publikasi atau berlangganan dapat gagal setelah dibuat, kemungkinan besar karena kesalahan jaringan. Kode kesalahan untuk “koneksi rekan hilang karena kesalahan jaringan” adalah 1400.

Ini dilaporkan secara asinkron melalui perender tahap yang disediakan aplikasi.

SDK mencoba ulang operasi terbitkan/berlangganan. Selama percobaan ulang, status terbitkan/berlangganan adalah/. `ATTEMPTING_PUBLISH` `ATTEMPTING_SUBSCRIBE` Jika upaya coba lagi berhasil, status berubah menjadi `PUBLISHED`/`SUBSCRIBED`.

SDK memanggil `onError` dengan kode kesalahan = 1400 dan fatal = false.

Tindakan: Tidak diperlukan tindakan, karena SDK mencoba ulang secara otomatis. Secara opsional, aplikasi dapat menyegarkan strategi untuk memaksa lebih banyak percobaan ulang. Jika terjadi kehilangan konektivitas total, kemungkinan koneksi ke Stages juga akan gagal.

SDK Siaran IVS: Panduan iOS (Streaming Waktu Nyata)

IVS real-time streaming iOS broadcast SDK memungkinkan peserta untuk mengirim dan menerima video di iOS.

`AmazonIVSBroadcastModul` mengimplementasikan antarmuka yang dijelaskan dalam dokumen ini. Operasi berikut didukung:

- Bergabunglah dengan panggung
- Publikasikan media ke peserta lain di panggung
- Berlangganan media dari peserta lain di panggung
- Kelola dan pantau video dan audio yang dipublikasikan ke panggung
- Dapatkan statistik WebRTC untuk setiap koneksi rekan
- Semua operasi dari SDK siaran iOS streaming latensi rendah IVS

Versi terbaru dari SDK siaran iOS: [1.14.1 \(Catatan Rilis\)](#)

Dokumentasi referensi: Untuk informasi tentang metode terpenting yang tersedia di SDK siaran Amazon IVS iOS, lihat dokumentasi referensi di <https://aws.github.io/amazon-ivs-broadcast-docs/1.14.1/ios/>.

Kode contoh: Lihat repositori contoh iOS di GitHub: <https://github.com/aws-samples/amazon-ivs-broadcast-ios-sample>.

Persyaratan platform: iOS 14 atau lebih tinggi

Memulai

Instal Perpustakaan

Kami menyarankan Anda mengintegrasikan SDK siaran melalui CocoaPods. (Atau, Anda dapat menambahkan kerangka kerja secara manual ke proyek Anda.)

Direkomendasikan: Integrasikan Broadcast SDK () CocoaPods

Fungsionalitas real-time diterbitkan sebagai subspec dari SDK siaran Streaming Latensi Rendah iOS. Ini agar pelanggan dapat memilih untuk memasukkan atau mengecualikannya berdasarkan kebutuhan fitur mereka. Termasuk itu meningkatkan ukuran paket.

Rilis diterbitkan melalui CocoaPods di bawah nama `AmazonIVSBroadcast`. Tambahkan dependensi ini ke Podfile Anda:

```
pod 'AmazonIVSBroadcast/Stages'
```

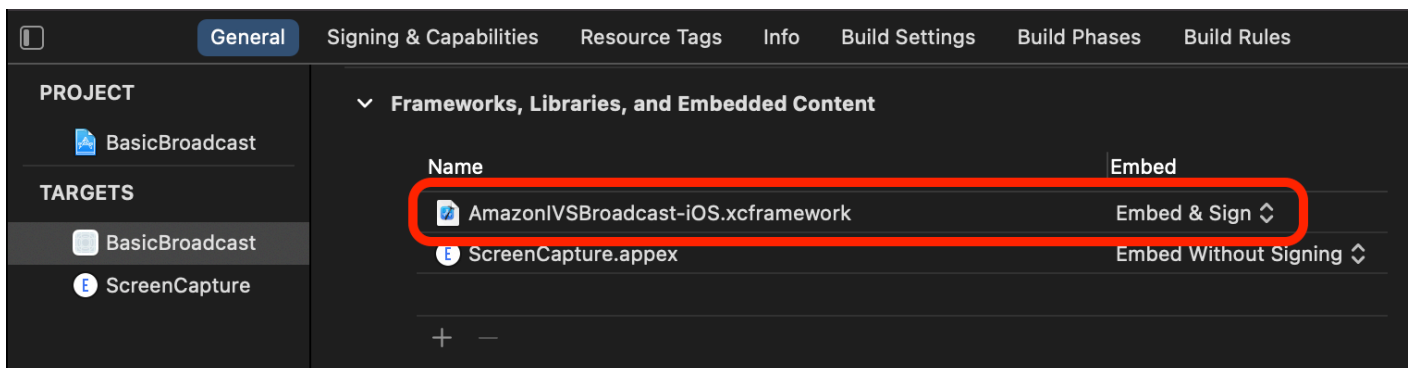
Jalankan `pod install` dan SDK akan tersedia di file `Anda.xcworkspace`.

Penting: SDK siaran streaming real-time IVS (yaitu, dengan subspec panggung) mencakup semua fitur SDK siaran streaming latensi rendah IVS. Tidak mungkin mengintegrasikan kedua SDK dalam proyek yang sama. Jika Anda menambahkan subspec stage via CocoaPods ke proyek Anda, pastikan untuk menghapus baris lain di Podfile yang berisi `AmazonIVSBroadcast`. Misalnya, tidak memiliki kedua baris ini di Podfile Anda:

```
pod 'AmazonIVSBroadcast'
pod 'AmazonIVSBroadcast/Stages'
```

Pendekatan Alternatif: Instal Kerangka Secara Manual

1. Unduh versi terbaru dari <https://broadcast.live-video.net/1.14.1/AmazonIVSBroadcast-Stages.xcframework.zip>.
2. Ekstrak isi arsip. `AmazonIVSBroadcast.xcframework` berisi SDK untuk perangkat dan simulator.
3. Sematkan `AmazonIVSBroadcast.xcframework` dengan menyeretnya ke bagian Frameworks, Libraries, dan Embedded Content pada tab General untuk target aplikasi Anda.



Permintaan Izin

Aplikasi Anda harus meminta izin untuk mengakses kamera dan mikrofon pengguna. (Ini tidak khusus untuk Amazon IVS; diperlukan untuk aplikasi apa pun yang membutuhkan akses ke kamera dan mikrofon.)

Di sini, kami memeriksa apakah pengguna telah memberikan izin dan, jika tidak, kami memintanya:

```
switch AVCaptureDevice.authorizationStatus(for: .video) {
case .authorized: // permission already granted.
case .notDetermined:
    AVCaptureDevice.requestAccess(for: .video) { granted in
        // permission granted based on granted bool.
    }
case .denied, .restricted: // permission denied.
@unknown default: // permissions unknown.
}
```

Anda perlu melakukan ini untuk keduanya `.video` dan jenis `.audio` media, jika Anda ingin akses ke kamera dan mikrofon, masing-masing.

Anda juga perlu menambahkan entri untuk `NSCameraUsageDescription` dan `NSMicrophoneUsageDescription` ke `AndaInfo.plist`. Jika tidak, aplikasi Anda akan macet saat mencoba meminta izin.

Nonaktifkan Aplikasi Idle Timer

Ini adalah langkah opsional, tetapi direkomendasikan. Ini mencegah perangkat Anda tertidur saat menggunakan SDK siaran, yang akan mengganggu siaran.

```
override func viewDidAppear(_ animated: Bool) {
    super.viewDidAppear(animated)
    UIApplication.shared.isIdleTimerDisabled = true
}
override func viewDidDisappear(_ animated: Bool) {
    super.viewDidDisappear(animated)
    UIApplication.shared.isIdleTimerDisabled = false
}
```

Penerbitan dan Berlangganan

Konsep

Tiga konsep inti mendasari fungsionalitas real-time: [panggung](#), [strategi](#), dan [penyaji](#). Tujuan desain adalah meminimalkan jumlah logika sisi klien yang diperlukan untuk membangun produk yang berfungsi.

Tahap

`IVSStageKelas` adalah titik utama interaksi antara aplikasi host dan SDK. Kelas mewakili panggung itu sendiri dan digunakan untuk bergabung dan meninggalkan panggung. Membuat atau bergabung dengan tahap memerlukan string token yang valid dan belum kedaluwarsa dari bidang kontrol (direpresentasikan sebagai token). Bergabung dan meninggalkan panggung itu sederhana.

```
let stage = try IVSStage(token: token, strategy: self)

try stage.join()

stage.leave()
```

`IVSStageKelas` juga adalah tempat `IVSStageRenderer` dan `IVSErrorDelegate` dapat dilampirkan:

```
let stage = try IVSStage(token: token, strategy: self)
stage.errorDelegate = self
stage.addRenderer(self) // multiple renderers can be added
```

Strategi

`IVSStageStrategyProtokol` menyediakan cara bagi aplikasi host untuk mengkomunikasikan keadaan tahap yang diinginkan ke SDK. Tiga fungsi perlu diimplementasikan: `shouldSubscribeToParticipant`, `shouldPublishParticipant`, dan `streamsToPublishForParticipant`. Semua dibahas di bawah ini.

Berlangganan Peserta

```
func stage(_ stage: IVSStage, shouldSubscribeToParticipant participant:
    IVSParticipantInfo) -> IVSStageSubscribeType
```

Ketika peserta jarak jauh bergabung dengan tahap, SDK menanyakan aplikasi host tentang status langganan yang diinginkan untuk peserta tersebut. Pilihannya adalah `.none`, `.audioOnly`, dan `.audioVideo`. Saat mengembalikan nilai untuk fungsi ini, aplikasi host tidak perlu khawatir tentang status publikasi, status langganan saat ini, atau status koneksi tahap. Jika `.audioVideo` dikembalikan, SDK menunggu hingga peserta jarak jauh memublikasikan sebelum berlangganan, dan memperbarui aplikasi host melalui perender selama proses berlangsung.

Berikut adalah contoh implementasi:

```
func stage(_ stage: IVSStage, shouldSubscribeToParticipant participant:
  IVSParticipantInfo) -> IVSStageSubscribeType {
    return .audioVideo
  }
```

Ini adalah implementasi lengkap dari fungsi ini untuk aplikasi host yang selalu ingin semua peserta saling bertemu; misalnya, aplikasi obrolan video.

Implementasi yang lebih maju juga dimungkinkan. Gunakan `attributes` properti `IVSParticipantInfo` untuk berlangganan peserta secara selektif berdasarkan atribut yang disediakan server:

```
func stage(_ stage: IVSStage, shouldSubscribeToParticipant participant:
  IVSParticipantInfo) -> IVSStageSubscribeType {
    switch participant.attributes["role"] {
    case "moderator": return .none
    case "guest": return .audioVideo
    default: return .none
    }
  }
```

Ini dapat digunakan untuk membuat panggung di mana moderator dapat memantau semua tamu tanpa terlihat atau didengar sendiri. Aplikasi host dapat menggunakan logika bisnis tambahan untuk membiarkan moderator melihat satu sama lain tetapi tetap tidak terlihat oleh tamu.

Penerbitan

```
func stage(_ stage: IVSStage, shouldPublishParticipant participant: IVSParticipantInfo)
  -> Bool
```

Setelah terhubung ke panggung, SDK menanyakan aplikasi host untuk melihat apakah peserta tertentu harus mempublikasikannya. Ini hanya dipanggil pada peserta lokal yang memiliki izin untuk mempublikasikan berdasarkan token yang disediakan.

Berikut adalah contoh implementasi:

```
func stage(_ stage: IVSStage, shouldPublishParticipant participant: IVSParticipantInfo)
    -> Bool {
    return true
}
```

Ini untuk aplikasi obrolan video standar di mana pengguna selalu ingin mempublikasikan. Mereka dapat membisukan dan menonaktifkan audio dan video mereka, untuk langsung disembunyikan atau dilihat/didengar. (Mereka juga dapat menggunakan publish/unpublish, tetapi itu jauh lebih lambat. Mute/unmute lebih disukai untuk kasus penggunaan di mana perubahan visibilitas sering diinginkan.)

Memilih Streaming untuk Publikasikan

```
func stage(_ stage: IVSStage, streamsToPublishForParticipant participant:
    IVSParticipantInfo) -> [IVSLocalStageStream]
```

Saat menerbitkan, ini digunakan untuk menentukan aliran audio dan video apa yang harus dipublikasikan. Ini dibahas secara lebih rinci nanti di [Publish a Media Stream](#).

Memperbarui Strategi

Strategi ini dimaksudkan untuk menjadi dinamis: nilai yang dikembalikan dari salah satu fungsi di atas dapat diubah kapan saja. Misalnya, jika aplikasi host tidak ingin mempublikasikan sampai pengguna akhir mengetuk tombol, Anda dapat mengembalikan variabel dari `shouldPublishParticipant` (sesuatu seperti `hasUserTappedPublishButton`). Ketika variabel itu berubah berdasarkan interaksi oleh pengguna akhir, panggil `stage.refreshStrategy()` untuk memberi sinyal ke SDK bahwa ia harus menanyakan strategi untuk nilai terbaru, hanya menerapkan hal-hal yang telah berubah. Jika SDK mengamati bahwa `shouldPublishParticipant` nilainya telah berubah, SDK akan memulai proses publikasi. Jika kueri SDK dan semua fungsi mengembalikan nilai yang sama seperti sebelumnya, `refreshStrategy` panggilan tidak akan membuat modifikasi apa pun pada tahapan.

Jika nilai pengembalian `shouldSubscribeToParticipant` perubahan dari `.audioVideo` ke `.audioOnly`, aliran video akan dihapus untuk semua peserta dengan nilai yang dikembalikan diubah, jika aliran video ada sebelumnya.

Umumnya, tahap menggunakan strategi untuk menerapkan perbedaan antara strategi sebelumnya dan saat ini secara efisien, tanpa aplikasi host perlu khawatir tentang semua keadaan yang diperlukan untuk mengelolanya dengan benar. Karena itu, anggap menelepon `stage.refreshStrategy()` sebagai operasi yang murah, karena tidak melakukan apa-apa kecuali strateginya berubah.

Penyaji

`IVSStageRenderer` protokol mengkomunikasikan keadaan panggung ke aplikasi host. Pembaruan pada UI aplikasi host biasanya dapat didukung sepenuhnya oleh peristiwa yang disediakan oleh perender. Penyaji menyediakan fungsi-fungsi berikut:

```
func stage(_ stage: IVSStage, participantDidJoin participant: IVSParticipantInfo)

func stage(_ stage: IVSStage, participantDidLeave participant: IVSParticipantInfo)

func stage(_ stage: IVSStage, participant: IVSParticipantInfo, didChange publishState:
  IVSParticipantPublishState)

func stage(_ stage: IVSStage, participant: IVSParticipantInfo, didChange
  subscribeState: IVSParticipantSubscribeState)

func stage(_ stage: IVSStage, participant: IVSParticipantInfo, didAdd streams:
  [IVSStageStream])

func stage(_ stage: IVSStage, participant: IVSParticipantInfo, didRemove streams:
  [IVSStageStream])

func stage(_ stage: IVSStage, participant: IVSParticipantInfo, didChangeMutedStreams
  streams: [IVSStageStream])

func stage(_ stage: IVSStage, didChange connectionState: IVSStageConnectionState,
  withError error: Error?)
```

Tidak diharapkan bahwa informasi yang diberikan oleh penyaji berdampak pada nilai pengembalian strategi. Misalnya, nilai pengembalian `shouldSubscribeToParticipant` tidak diharapkan berubah ketika `participant:didChangePublishState` dipanggil. Jika aplikasi host ingin berlangganan ke peserta tertentu, itu harus mengembalikan jenis langganan yang diinginkan terlepas dari status publikasi peserta tersebut. SDK bertanggung jawab untuk memastikan bahwa keadaan strategi yang diinginkan ditindaklanjuti pada waktu yang tepat berdasarkan keadaan tahap.

Perhatikan bahwa hanya peserta penerbitan yang dipicu `participantDidJoin`, dan setiap kali peserta berhenti menerbitkan atau meninggalkan sesi panggung, `participantDidLeave` dipicu.

Publikasikan Aliran Media

Perangkat lokal seperti mikrofon dan kamera internal ditemukan melalui `IVSDeviceDiscovery`. Berikut adalah contoh memilih kamera yang menghadap ke depan dan mikrofon default, lalu mengembalikannya `IVSLocalStageStreams` agar dipublikasikan oleh SDK:

```
let devices = IVSDeviceDiscovery.listLocalDevices()

// Find the camera virtual device, choose the front source, and create a stream
let camera = devices.compactMap({ $0 as? IVSCamera }).first!
let frontSource = camera.listAvailableInputSources().first(where: { $0.position == .front })!
camera.setPreferredInputSource(frontSource)
let cameraStream = IVSLocalStageStream(device: camera)

// Find the microphone virtual device, enable echo cancellation, and create a stream
let microphone = devices.compactMap({ $0 as? IVSMicrophone }).first!
microphone.isEchoCancellationEnabled = true
let microphoneStream = IVSLocalStageStream(device: microphone)

// This is a function on IVSStageStrategy
func stage(_ stage: IVSStage, streamsToPublishForParticipant participant:
  IVSParticipantInfo) -> [IVSLocalStageStream] {
    return [cameraStream, microphoneStream]
  }
```

Tampilkan dan Hapus Peserta

Setelah berlangganan selesai, Anda akan menerima array `IVSStageStream` objek melalui fungsi `renderer.didAddStreams` Untuk melihat pratinjau atau menerima statistik tingkat audio tentang peserta ini, Anda dapat mengakses `IVSDevice` objek yang mendasarinya dari aliran:

```
if let imageDevice = stream.device as? IVSImageDevice {
    let preview = imageDevice.previewView()
    /* attach this UIView subclass to your view */
} else if let audioDevice = stream.device as? IVSAudioDevice {
    audioDevice.setStatsCallback( { stats in
        /* process stats.peak and stats.rms */
    })
}
```

```
}
```

Ketika peserta berhenti menerbitkan atau berhenti berlangganan, `didRemoveStreams` fungsi dipanggil dengan aliran yang telah dihapus. Aplikasi host harus menggunakan ini sebagai sinyal untuk menghapus aliran video peserta dari hierarki tampilan.

`didRemoveStreams` dipanggil untuk semua skenario di mana aliran mungkin dihapus, termasuk:

- Peserta jarak jauh berhenti menerbitkan.
- Perangkat lokal berhenti berlangganan atau mengubah langganan dari ke `.audioVideo`.
`.audioOnly`
- Peserta jarak jauh meninggalkan panggung.
- Peserta lokal meninggalkan panggung.

Karena `didRemoveStreams` dipanggil untuk semua skenario, tidak diperlukan logika bisnis khusus untuk menghapus peserta dari UI selama operasi cuti jarak jauh atau lokal.

Bisukan dan Bunyikan Streaming Media

`IVSLocalStageStream` objek memiliki `setMuted` fungsi yang mengontrol apakah aliran diredam. Fungsi ini dapat dipanggil pada aliran sebelum atau sesudah dikembalikan dari fungsi `streamsToPublishForParticipant` strategi.

Penting: Jika instance `IVSLocalStageStream` objek baru dikembalikan `streamsToPublishForParticipant` setelah panggilan `refreshStrategy`, status bisu objek aliran baru diterapkan ke panggung. Hati-hati saat membuat `IVSLocalStageStream` instance baru untuk memastikan status bisu yang diharapkan dipertahankan.

Pantau Status Bisu Media Peserta Jarak Jauh

Saat peserta mengubah status bisu aliran video atau audionya, `didChangeMutedStreams` fungsi penyaji dipanggil dengan larik aliran yang telah berubah. Gunakan `isMuted` properti `IVSStageStream` untuk memperbarui UI Anda sesuai:

```
func stage(_ stage: IVSStage, participant: IVSParticipantInfo, didChangeMutedStreams
  streams: [IVSStageStream]) {
  streams.forEach { stream in
    /* stream.isMuted */
  }
}
```

```
}

```

Buat Konfigurasi Panggung

Untuk menyesuaikan nilai konfigurasi video panggung, gunakan `IVSLocalStageStreamVideoConfiguration`:

```
let config = IVSLocalStageStreamVideoConfiguration()
try config.setMaxBitrate(900_000)
try config.setMinBitrate(100_000)
try config.setTargetFramerate(30)
try config.setSize(CGSize(width: 360, height: 640))
config.degradationPreference = .balanced

```

Dapatkan Statistik WebRTC

Untuk mendapatkan statistik WebRTC terbaru untuk aliran penerbitan atau aliran berlangganan, gunakan terus `requestRTCStats` `IVSStageStream` Ketika koleksi selesai, Anda akan menerima statistik melalui `IVSStageStreamDelegate` yang dapat diatur `IVSStageStream`. Untuk terus mengumpulkan statistik WebRTC, panggil fungsi ini di file. Timer

```
func stream(_ stream: IVSStageStream, didGenerateRTCStats stats: [String : [String : String]]) {
    for stat in stats {
        for member in stat.value {
            print("stat \(stat.key) has member \(member.key) with value \(member.value)")
        }
    }
}

```

Dapatkan Atribut Peserta

Jika Anda menentukan atribut dalam permintaan `CreateParticipantToken` titik akhir, Anda dapat melihat atribut di `IVSParticipantInfo` properti:

```
func stage(_ stage: IVSStage, participantDidJoin participant: IVSParticipantInfo) {
    print("ID: \(participant.participantId)")
    for attribute in participant.attributes {
        print("attribute: \(attribute.key)=\(attribute.value)")
    }
}

```

Lanjutkan Sesi di Latar Belakang

Saat aplikasi memasuki latar belakang, Anda dapat terus berada di panggung sambil mendengar audio jarak jauh, meskipun tidak mungkin untuk terus mengirim gambar dan audio Anda sendiri. Anda perlu memperbarui `IVSStageStrategy` implementasi Anda untuk berhenti menerbitkan dan berlangganan `.audioOnly` (atau `.none`, jika ada):

```
func stage(_ stage: IVSStage, shouldPublishParticipant participant: IVSParticipantInfo)
    -> Bool {
    return false
}
func stage(_ stage: IVSStage, shouldSubscribeToParticipant participant:
    IVSParticipantInfo) -> IVSStageSubscribeType {
    return .audioOnly
}
```

Kemudian buat panggilan `kestage.refreshStrategy()`.

Aktifkan/Nonaktifkan Layered Encoding dengan Simulcast

Saat menerbitkan aliran media, SDK mentransmisikan streaming video berkualitas tinggi dan berkualitas rendah, sehingga peserta jarak jauh dapat berlangganan streaming meskipun mereka memiliki bandwidth downlink terbatas. Pengkodean berlapis dengan simulcast aktif secara default. Anda dapat menonaktifkannya dengan `IVSSimulcastConfiguration`:

```
// Disable Simulcast
let config = IVSLocalStageStreamVideoConfiguration()
config.simulcast.enabled = false

let cameraStream = IVSLocalStageStream(device: camera, configuration: config)

// Other Stage implementation code
```

Siarkan Panggung ke Saluran IVS

Untuk menyiarkan panggung, buat yang terpisah `IVSBroadcastSession` dan kemudian ikuti instruksi biasa untuk penyiaran dengan SDK, yang dijelaskan di atas. `deviceProperti` pada `IVSStageStream` akan berupa `IVSImageDevice` atau `IVSAudioDevice` seperti yang ditunjukkan pada cuplikan di atas; ini dapat dihubungkan ke `IVSBroadcastSession.mixer` untuk menyiarkan seluruh tahap dalam tata letak yang dapat disesuaikan.

Secara opsional, Anda dapat menggabungkan panggung dan menyiarkannya ke saluran latensi rendah IVS, untuk menjangkau audiens yang lebih besar. Lihat [Mengaktifkan Beberapa Host di Amazon IVS Stream di Panduan Pengguna Streaming](#) Latensi Rendah IVS.

Bagaimana iOS Memilih Resolusi Kamera dan Frame Rate

Kamera yang dikelola oleh SDK siaran mengoptimalkan resolusi dan frame rate (frames-per-second, atau FPS) untuk meminimalkan produksi panas dan konsumsi energi. Bagian ini menjelaskan bagaimana resolusi dan frame rate dipilih untuk membantu aplikasi host mengoptimalkan kasus penggunaannya.

Saat membuat `IVSLocalStageStream` dengan `IVSCamera`, kamera dioptimalkan untuk frame rate `IVSLocalStageStreamVideoConfiguration.targetFramerate` dan resolusi `IVSLocalStageStreamVideoConfiguration.size`. Panggilan `IVSLocalStageStream.setConfiguration` memperbarui kamera dengan nilai yang lebih baru.

Pratinjau Kamera

Jika Anda membuat pratinjau `IVSCamera` tanpa melampirkannya ke `IVSBroadcastSession` atau `IVSStage`, defaultnya ke resolusi 1080p dan frame rate 60 fps.

Menyiarkan Panggung

Saat menggunakan `IVSBroadcastSession` untuk menyiarkan `IVSStage`, SDK mencoba mengoptimalkan kamera dengan resolusi dan kecepatan bingkai yang memenuhi kriteria kedua sesi.

Misalnya, jika konfigurasi siaran diatur untuk memiliki frame rate 15 FPS dan resolusi 1080p, sedangkan Stage memiliki frame rate 30 FPS dan resolusi 720p, SDK akan memilih konfigurasi kamera dengan frame rate 30 FPS dan resolusi 1080p. Ini `IVSBroadcastSession` akan menjatuhkan setiap bingkai lain dari kamera, dan akan `IVSStage` menskalakan gambar 1080p ke 720p.

Jika aplikasi host berencana menggunakan keduanya `IVSBroadcastSession` dan `IVSStage` bersama-sama, dengan kamera, kami menyarankan agar `targetFramerate` dan `size` properti dari konfigurasi masing-masing cocok. Ketidakcocokan dapat menyebabkan kamera mengkonfigurasi ulang dirinya sendiri saat merekam video, yang akan menyebabkan penundaan singkat dalam pengiriman sampel video.

Jika memiliki nilai yang identik tidak memenuhi kasus penggunaan aplikasi host, membuat kamera berkualitas lebih tinggi terlebih dahulu akan mencegah kamera mengkonfigurasi ulang dirinya sendiri

ketika sesi kualitas yang lebih rendah ditambahkan. Misalnya, jika Anda menyiarkan pada 1080p dan 30 FPS dan kemudian bergabung dengan Stage yang disetel ke 720p dan 30 FPS, kamera tidak akan mengkonfigurasi ulang dirinya sendiri dan video akan terus tanpa gangguan. Ini karena 720p kurang dari atau sama dengan 1080p dan 30 FPS kurang dari atau sama dengan 30 FPS.

Frame Rate, Resolusi, dan Rasio Aspek Sewenang-wenang

Sebagian besar perangkat keras kamera dapat sama persis dengan format umum, seperti 720p pada 30 FPS atau 1080p pada 60 FPS. Namun, tidak mungkin untuk benar-benar mencocokkan semua format. SDK siaran memilih konfigurasi kamera berdasarkan aturan berikut (dalam urutan prioritas):

1. Lebar dan tinggi resolusi lebih besar dari atau sama dengan resolusi yang diinginkan, tetapi dalam batasan ini, lebar dan tinggi sekecil mungkin.
2. Frame rate lebih besar dari atau sama dengan frame rate yang diinginkan, tetapi dalam batasan ini, frame rate serendah mungkin.
3. Rasio aspek cocok dengan rasio aspek yang diinginkan.
4. Jika ada beberapa format yang cocok, format dengan bidang pandang terbesar digunakan.

Berikut adalah dua contoh:

- Aplikasi host mencoba menyiarkan dalam 4k pada 120 FPS. Kamera yang dipilih hanya mendukung 4k pada 60 FPS atau 1080p pada 120 FPS. Format yang dipilih akan menjadi 4k pada 60 FPS, karena aturan resolusi lebih tinggi prioritas daripada aturan frame-rate.
- Resolusi tidak teratur diminta, 1910x1070. Kamera akan menggunakan 1920x1080. Hati-hati: memilih resolusi seperti 1921x1080 akan menyebabkan kamera meningkatkan ke resolusi berikutnya yang tersedia (seperti 2592x1944), yang menimbulkan penalti CPU dan memory-bandwidth.

Bagaimana dengan Android?

Android tidak menyesuaikan resolusi atau frame rate dengan cepat seperti iOS, jadi ini tidak memengaruhi SDK siaran Android.

Masalah dan Solusi yang Diketahui

- Mengubah rute audio Bluetooth tidak dapat diprediksi. Jika Anda menghubungkan perangkat baru di tengah sesi, iOS mungkin atau mungkin tidak secara otomatis mengubah rute input. Selain itu,

tidak mungkin memilih di antara beberapa headset Bluetooth yang terhubung secara bersamaan. Ini terjadi di sesi siaran reguler dan panggung.

Solusi: Jika Anda berencana untuk menggunakan headset Bluetooth, sambungkan sebelum memulai siaran atau panggung dan biarkan terhubung sepanjang sesi.

- Peserta yang menggunakan iPhone 14, iPhone 14 Plus, iPhone 14 Pro, atau iPhone 14 Pro Max dapat menyebabkan masalah gema audio bagi peserta lain.

Solusi: Peserta yang menggunakan perangkat yang terpengaruh dapat menggunakan headphone untuk mencegah masalah gema bagi peserta lain.

- Ketika peserta bergabung dengan token yang digunakan oleh peserta lain, koneksi pertama terputus tanpa kesalahan tertentu.

Solusi: Tidak ada.

- Ada masalah langka di mana penerbit menerbitkan tetapi status publikasi yang diterima pelanggan adalah `inactive`.

Solusi: Coba pergi dan kemudian bergabung dengan sesi. Jika masalah tetap ada, buat token baru untuk penerbit.

- Ketika seorang peserta menerbitkan atau berlangganan, dimungkinkan untuk menerima kesalahan dengan kode 1400 yang menunjukkan pemutusan karena masalah jaringan, bahkan ketika jaringan stabil.

Solusi: Coba terbitkan ulang/berlangganan ulang.

- Masalah distorsi audio yang jarang terjadi dapat terjadi sebentar-sebentar selama sesi panggung, biasanya pada panggilan dengan durasi yang lebih lama.

Solusi: Peserta dengan audio yang terdistorsi dapat meninggalkan dan bergabung kembali dengan sesi, atau membatalkan publikasi dan menerbitkan ulang audio mereka untuk memperbaiki masalah.

Penanganan Kesalahan

Kesalahan Fatal vs Non-Fatal

Objek kesalahan memiliki boolean “fatal”. Ini adalah entri kamus di bawahnya `IVSBroadcastErrorIsFatalKey` yang berisi boolean.

Secara umum, kesalahan fatal terkait dengan koneksi ke server Tahapan (baik koneksi tidak dapat dibuat atau hilang dan tidak dapat dipulihkan). Aplikasi harus membuat ulang panggung dan bergabung kembali, mungkin dengan token baru atau ketika konektivitas perangkat pulih.

Kesalahan non-fatal umumnya terkait dengan status terbitkan/berlangganan dan ditangani oleh SDK, yang mencoba kembali operasi terbitkan/berlangganan.

Anda dapat memeriksa properti ini:

```
let nsError = error as NSError
if nsError.userInfo[IVSBroadcastErrorIsFatalKey] as? Bool == true {
    // the error is fatal
}
```

Bergabung Kesalahan

Token Cacat

Ini terjadi ketika token panggung salah bentuk.

SDK melempar pengecualian Swift dengan kode kesalahan = 1000 dan BroadcastErrorIsFatalKey IVS = YA.

Tindakan: Buat token yang valid dan coba lagi bergabung.

Token Kadaluwarsa

Ini terjadi ketika token panggung kedaluwarsa.

SDK melempar pengecualian Swift dengan kode kesalahan = 1001 dan IVS = YA.
BroadcastErrorIsFatalKey

Tindakan: Buat token baru dan coba lagi bergabung.

Token Tidak Valid atau Dicabut

Ini terjadi ketika token panggung tidak cacat tetapi ditolak oleh server Stages. Ini dilaporkan secara asinkron melalui perender tahap yang disediakan aplikasi.

Panggilan SDK `stage(didChange connectionState, withError error)` dengan kode kesalahan = 1026 dan IVS BroadcastErrorIsFatalKey = YA.

Tindakan: Buat token yang valid dan coba lagi bergabung.

Kesalahan Jaringan untuk Gabung Awal

Ini terjadi ketika SDK tidak dapat menghubungi server Stages untuk membuat koneksi. Ini dilaporkan secara asinkron melalui perender tahap yang disediakan aplikasi.

Panggilan SDK `stage(didChange connectionState, withError error)` dengan kode kesalahan = 1300 dan `IVS BroadcastErrorIsFatalKey = YA`.

Tindakan: Tunggu konektivitas perangkat pulih dan coba lagi bergabung.

Kesalahan Jaringan saat Sudah Bergabung

Jika koneksi jaringan perangkat mati, SDK mungkin kehilangan koneksinya ke server Stage. Ini dilaporkan secara asinkron melalui perender tahap yang disediakan aplikasi.

Panggilan SDK `stage(didChange connectionState, withError error)` dengan kode kesalahan = 1300 dan `BroadcastErrorIsFatalKey` nilai IVS = YA.

Tindakan: Tunggu konektivitas perangkat pulih dan coba lagi bergabung.

Kesalahan Publikasi/Berlangganan

Awal

Ada beberapa kesalahan:

- `MultihostSessionOfferCreationFailPublish` (1020)
- `MultihostSessionOfferCreationFailSubscribe` (1021)
- `MultihostSessionNolceCandidates` (1022)
- `MultihostSessionStageAtCapacity` (1024)
- `SignallingSessionCannotRead` (1201)
- `SignallingSessionCannotSend` (1202)
- `SignallingSessionBadResponse` (1203)

Ini dilaporkan secara asinkron melalui perender tahap yang disediakan aplikasi.

SDK mencoba ulang operasi untuk beberapa kali. Selama percobaan ulang, status terbitkan/berlangganan adalah/. `ATTEMPTING_PUBLISH` `ATTEMPTING_SUBSCRIBE` Jika upaya coba lagi berhasil, status berubah menjadi `PUBLISHED`/`SUBSCRIBED`.

Panggilan SDK `IVSErrorSourceDelegate:didEmitError` dengan kode kesalahan yang relevan dan `IVS BroadcastErrorIsFatalKey = NO`.

Tindakan: Tidak diperlukan tindakan, karena SDK mencoba ulang secara otomatis. Secara opsional, aplikasi dapat menyegarkan strategi untuk memaksa lebih banyak percobaan ulang.

Sudah Didirikan, Lalu Gagal

Publikasi atau berlangganan dapat gagal setelah dibuat, kemungkinan besar karena kesalahan jaringan. Kode kesalahan untuk “koneksi rekan hilang karena kesalahan jaringan” adalah 1400.

Ini dilaporkan secara asinkron melalui perender tahap yang disediakan aplikasi.

SDK mencoba ulang operasi terbitkan/berlangganan. Selama percobaan ulang, status terbitkan/berlangganan adalah/. `ATTEMPTING_PUBLISH` `ATTEMPTING_SUBSCRIBE` Jika upaya coba lagi berhasil, status berubah menjadi `PUBLISHED`/`SUBSCRIBED`.

Panggilan SDK `didEmitError` dengan kode kesalahan = 1400 dan `IVS BroadcastErrorIsFatalKey = NO`.

Tindakan: Tidak diperlukan tindakan, karena SDK mencoba ulang secara otomatis. Secara opsional, aplikasi dapat menyegarkan strategi untuk memaksa lebih banyak percobaan ulang. Jika terjadi kehilangan konektivitas total, kemungkinan koneksi ke Stages juga akan gagal.

SDK Siaran IVS: Sumber Gambar Khusus (Streaming Real-Time)

Sumber input gambar khusus memungkinkan aplikasi menyediakan input gambarnya sendiri ke SDK siaran, alih-alih dibatasi pada kamera preset. Sumber gambar khusus bisa sesederhana tanda air semi-transparan atau adegan statis “be right back”, atau dapat memungkinkan aplikasi untuk melakukan pemrosesan khusus tambahan seperti menambahkan filter kecantikan ke kamera.

Bila Anda menggunakan sumber input gambar khusus untuk kontrol kustom kamera (seperti menggunakan pustaka filter kecantikan yang memerlukan akses kamera), SDK siaran tidak lagi bertanggung jawab untuk mengelola kamera. Sebagai gantinya, aplikasi bertanggung jawab untuk menangani siklus hidup kamera dengan benar. Lihat dokumentasi platform resmi tentang bagaimana aplikasi Anda mengelola kamera.

Android

Setelah Anda membuat `DeviceDiscovery` sesi, membuat sumber gambar-masukan:

```
CustomImageSource imageSource = deviceDiscovery.createImageInputSource(new  
BroadcastConfiguration.Vec2(1280, 720));
```

Metode ini mengembalikan `CustomImageSource`, yang merupakan sumber gambar yang didukung oleh Android standar [Permukaan](#). Subclass `SurfaceSource` dapat diubah ukurannya dan diputar. Anda juga dapat membuat `ImagePreviewView` untuk menampilkan pratinjau isinya.

Untuk mengambil yang mendasari `Surface`:

```
Surface surface = surfaceSource.getInputSurface();
```

`IniSurface` dapat digunakan sebagai buffer keluaran untuk produsen gambar seperti `Camera2`, `OpenGL ES`, dan pustaka lainnya. Kasus penggunaan yang paling sederhana adalah langsung menggambar bitmap statis atau warna ke `Canvas Surface`. Namun, banyak pustaka (seperti pustaka filter kecantikan) menyediakan metode yang memungkinkan aplikasi untuk menentukan eksternal `Surface` untuk rendering. Anda dapat menggunakan metode seperti itu untuk melewati `IniSurface` ke pustaka filter, yang memungkinkan perpustakaan untuk menampilkan frame yang diproses untuk sesi siaran untuk streaming.

`IniCustomImageSource` dapat dibungkus dalam `LocalStageStream` dan dikembalikan oleh `StageStrategy` untuk mempublikasikan ke `Stage`.

iOS

Setelah Anda membuat `DeviceDiscovery` sesi, membuat sumber gambar-masukan:

```
let customSource = broadcastSession.createImageSource(withName: "customSourceName")
```

Metode ini mengembalikan `IVSCustomImageSource`, yang merupakan sumber gambar yang memungkinkan aplikasi untuk mengirimkan `CMSampleBuffer` secara manual. Untuk format piksel yang didukung, lihat Referensi SDK Siaran iOS; tautan ke versi terbaru ada di [Catatan Rilis Amazon IVS](#) untuk rilis SDK siaran terbaru.

Sampel yang dikirimkan ke sumber khusus akan dialirkan ke Tahap:

```
customSource.onSampleBuffer(sampleBuffer)
```

Untuk streaming video, gunakan metode ini dalam callback. Misalnya, jika Anda menggunakan kamera, maka setiap kali buffer sampel baru diterima dari sebuah `AVCaptureSession`, aplikasi dapat meneruskan buffer sampel ke sumber gambar khusus. Jika diinginkan, aplikasi dapat menerapkan pemrosesan lebih lanjut (seperti filter kecantikan) sebelum mengirimkan sampel ke sumber gambar khusus.

Yang `IVSCustomImageSource` dapat dibungkus dalam `IVSLocalStageStream` dan dikembalikan oleh `IVSStageStrategy` untuk mempublikasikan ke `Stage`.

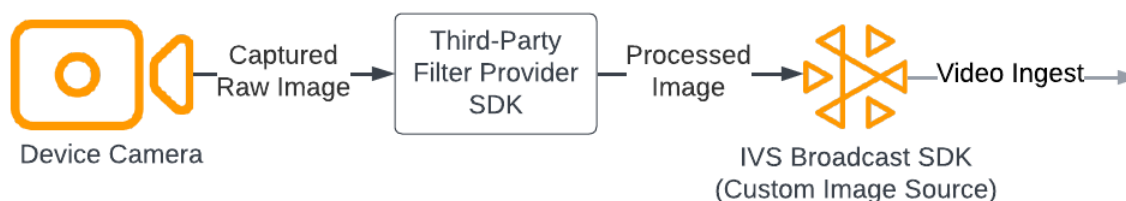
SDK Siaran IVS: Filter Kamera Pihak Ketiga (Streaming Waktu Nyata)

Panduan ini mengasumsikan Anda sudah terbiasa dengan sumber [gambar khusus](#) serta mengintegrasikan [SDK siaran streaming real-time IVS](#) ke dalam aplikasi Anda.

Filter kamera memungkinkan pembuat live-stream untuk menambah atau mengubah tampilan wajah atau latar belakang mereka. Ini berpotensi meningkatkan keterlibatan pemirsa, menarik pemirsa, dan meningkatkan pengalaman streaming langsung.

Mengintegrasikan Filter Kamera Pihak Ketiga

Anda dapat mengintegrasikan SDK filter kamera pihak ketiga dengan SDK siaran IVS dengan memasukkan output SDK filter ke sumber input gambar [khusus](#). Sumber input gambar khusus memungkinkan aplikasi memberikan input gambarnya sendiri ke Broadcast SDK. SDK penyedia filter pihak ketiga dapat mengelola siklus hidup kamera untuk memproses gambar dari kamera, menerapkan efek filter, dan mengeluarkannya dalam format yang dapat diteruskan ke sumber gambar khusus.



Konsultasikan dokumentasi penyedia filter pihak ketiga Anda untuk mengetahui metode bawaan untuk mengonversi bingkai kamera, dengan efek filter, diterapkan ke format yang dapat diteruskan

ke sumber [input gambar khusus](#). Prosesnya bervariasi, tergantung pada versi SDK siaran IVS mana yang digunakan:

- Web — Penyedia filter harus dapat merender outputnya ke elemen kanvas. Metode [captureStream](#) kemudian dapat digunakan untuk mengembalikan `MediaStream` konten kanvas. Kemudian `MediaStream` dapat dikonversi ke instance [LocalStageStream](#) dan diterbitkan ke Tahap.
- Android — SDK penyedia filter dapat merender bingkai ke Android yang `Surface` disediakan oleh SDK siaran IVS atau mengonversi bingkai menjadi bitmap. Jika menggunakan bitmap, itu kemudian dapat dirender ke dasar yang `Surface` disediakan oleh sumber gambar kustom, dengan membuka kunci dan menulis ke kanvas.
- iOS — SDK penyedia filter pihak ketiga harus menyediakan bingkai kamera dengan efek filter yang diterapkan sebagai `CMSampleBuffer` file. Lihat dokumentasi SDK vendor filter pihak ketiga Anda untuk informasi tentang cara mendapatkan `CMSampleBuffer` hasil akhir setelah gambar kamera diproses.

BytePlus

Android

Instal dan Atur BytePlus Efek SDK

Lihat [Panduan Akses BytePlus Android](#) untuk detail tentang cara menginstal, menginisialisasi, dan menyiapkan BytePlus Effects SDK.

Mengatur Sumber Gambar Kustom

Setelah menginisialisasi SDK, berikan bingkai kamera yang diproses dengan efek filter yang diterapkan ke sumber input gambar khusus. Untuk melakukan itu, buat instance `DeviceDiscovery` objek dan buat sumber gambar khusus. Perhatikan bahwa saat Anda menggunakan sumber input gambar khusus untuk kontrol kustom kamera, SDK siaran tidak lagi bertanggung jawab untuk mengelola kamera. Sebagai gantinya, aplikasi bertanggung jawab untuk menangani siklus hidup kamera dengan benar.

Java

```
var deviceDiscovery = DeviceDiscovery(applicationContext)
var customSource = deviceDiscovery.createImageInputSource( BroadcastConfiguration.Vec2(
    720F, 1280F
```

```

))
var surface: Surface = customSource.inputSurface
var filterStream = ImageLocalStageStream(customSource)

```

Konversi Output ke Bitmap dan Umpan ke Sumber Input Gambar Kustom

Untuk mengaktifkan bingkai kamera dengan efek filter yang diterapkan dari BytePlus Effect SDK untuk diteruskan langsung ke SDK siaran IVS, ubah output tekstur BytePlus Effects SDK menjadi bitmap. Saat gambar diproses, `onDrawFrame()` metode ini dipanggil oleh SDK. `onDrawFrame()` Metode ini adalah metode publik antarmuka [GL SurfaceView.Renderer](#) Android. Di aplikasi sampel Android yang disediakan oleh BytePlus, metode ini dipanggil pada setiap bingkai kamera; ini menghasilkan tekstur. Secara bersamaan, Anda dapat melengkapi `onDrawFrame()` metode dengan logika untuk mengonversi tekstur ini menjadi bitmap dan memasukkannya ke sumber input gambar khusus. Seperti yang ditunjukkan pada contoh kode berikut, gunakan `transferTextureToBitmap` metode yang disediakan oleh BytePlus SDK untuk melakukan konversi ini. Metode ini disediakan oleh [com.bytedance.labcv.core.util.ImageUtil](#) dari BytePlus Effects SDK, seperti yang ditunjukkan pada contoh kode berikut. Anda kemudian dapat merender ke Android Surface dasar a CustomImageSource dengan menulis bitmap yang dihasilkan ke kanvas Surface. Banyak pemanggilan berturut-turut `onDrawFrame()` menghasilkan urutan bitmap, dan ketika digabungkan, menciptakan aliran video.

Java

```

import com.bytedance.labcv.core.util.ImageUtil;
...
protected ImageUtil imageUtility;
...

@Override
public void onDrawFrame(GL10 gl10) {
    ...
    // Convert BytePlus output to a Bitmap
    Bitmap outputBt = imageUtility.transferTextureToBitmap(output.getTexture(), ByteEffect
    Constants.TextureFormat.Texture2D, output.getWidth(), output.getHeight());

    canvas = surface.lockCanvas(null);
    canvas.drawBitmap(outputBt, 0f, 0f, null);
    surface.unlockCanvasAndPost(canvas);
}

```

DeepAR

Android

Lihat [Panduan Integrasi Android dari DeepAR](#) untuk detail tentang cara mengintegrasikan DeepAR SDK dengan SDK siaran Android IVS.

iOS

Lihat [Panduan Integrasi iOS dari DeepAR](#) untuk detail tentang cara mengintegrasikan DeepAR SDK dengan SDK siaran iOS IVS.

Jepret

Web

Bagian ini mengasumsikan Anda sudah terbiasa dengan [penerbitan dan berlangganan video menggunakan Web Broadcast SDK](#).

Untuk mengintegrasikan SDK Kit Kamera Snap dengan SDK siaran Web streaming real-time IVS, Anda perlu:

1. Instal Camera Kit SDK dan Webpack. (Contoh kami menggunakan Webpack sebagai bundler, tetapi Anda dapat menggunakan bundler pilihan Anda.)
2. Buat `index.html`.
3. Tambahkan elemen pengaturan.
4. Tampilkan dan atur peserta.
5. Tampilkan kamera dan mikrofon yang terhubung.
6. Buat sesi Kit Kamera.
7. Ambil dan terapkan Lensa.
8. Render output dari sesi Kit Kamera ke kanvas.
9. Menyediakan Kit Kamera dengan sumber media untuk rendering dan mempublikasikan file. `LocalStageStream`
10. Buat file konfigurasi Webpack.

Masing-masing langkah ini dijelaskan di bawah ini.

Instal SDK Kit Kamera dan Webpack

```
npm i @snap/camera-kit webpack webpack-cli
```

Buat index.html

Selanjutnya, buat boilerplate HTML dan impor SDK siaran Web sebagai tag skrip. Dalam kode berikut, pastikan untuk mengganti `<SDK version>` dengan versi SDK siaran yang Anda gunakan.

JavaScript

```
<!--
/*! Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved. SPDX-License-
Identifier: Apache-2.0 */
-->
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />

  <title>Amazon IVS Real-Time Streaming Web Sample (HTML and JavaScript)</title>

  <!-- Fonts and Styling -->
  <link rel="stylesheet" href="https://fonts.googleapis.com/css?
family=Roboto:300,300italic,700,700italic" />
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/normalize/8.0.1/
normalize.css" />
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/milligram/1.4.1/
milligram.css" />
  <link rel="stylesheet" href="./index.css" />

  <!-- Stages in Broadcast SDK -->
  <script src="https://web-broadcast.live-video.net/<SDK version>/amazon-ivs-web-
broadcast.js"></script>
</head>

<body>
  <!-- Introduction -->
  <header>
    <h1>Amazon IVS Real-Time Streaming Web Sample (HTML and JavaScript)</h1>
```



```

    <p>This sample is used to demonstrate basic HTML / JS usage. <b><a href="https://
docs.aws.amazon.com/ivs/latest/userguide/multiple-hosts.html">Use the AWS CLI</
a></b> to create a <b>Stage</b> and a corresponding <b>ParticipantToken</b>.
Multiple participants can load this page and put in their own tokens. You can <b><a
href="https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-guides/stages#glossary"
target="_blank">read more about stages in our public docs.</a></b></p>
</header>
<hr />

<!-- Setup Controls -->

<!-- Local Participant -->

<hr style="margin-top: 5rem"/>

<!-- Remote Participants -->

<!-- Load all Desired Scripts -->

</body>

</html>

```

Tambahkan Elemen Pengaturan

Buat HTML untuk memilih kamera dan mikrofon dan menentukan token peserta:

JavaScript

```

<!-- Setup Controls -->
<div class="row">
  <div class="column">
    <label for="video-devices">Select Camera</label>
    <select disabled id="video-devices">
      <option selected disabled>Choose Option</option>
    </select>
  </div>
  <div class="column">
    <label for="audio-devices">Select Microphone</label>
    <select disabled id="audio-devices">
      <option selected disabled>Choose Option</option>
    </select>
  </div>
</div>

```

```

</div>
<div class="column">
  <label for="token">Participant Token</label>
  <input type="text" id="token" name="token" />
</div>
<div class="column" style="display: flex; margin-top: 1.5rem">
  <button class="button" style="margin: auto; width: 100%" id="join-button">Join
Stage</button>
</div>
<div class="column" style="display: flex; margin-top: 1.5rem">
  <button class="button" style="margin: auto; width: 100%" id="leave-button">Leave
Stage</button>
</div>
</div>

```

Tambahkan HTML tambahan di bawahnya untuk menampilkan umpan kamera dari peserta lokal dan jarak jauh:

JavaScript

```

<!-- Local Participant -->
<div class="row local-container">
  <canvas id="canvas"></canvas>

  <div class="column" id="local-media"></div>
  <div class="static-controls hidden" id="local-controls">
    <button class="button" id="mic-control">Mute Mic</button>
    <button class="button" id="camera-control">Mute Camera</button>
  </div>
</div>

<hr style="margin-top: 5rem"/>

<!-- Remote Participants -->
<div class="row">
  <div id="remote-media"></div>
</div>

```

Muat logika tambahan, termasuk metode pembantu untuk mengatur kamera dan file yang dibundel JavaScript. (Nanti di bagian ini, Anda akan membuat JavaScript file-file ini dan menggabungkannya menjadi satu file, sehingga Anda dapat mengimpor Kit Kamera sebagai modul. JavaScript File yang

dibundel akan berisi logika untuk menyiapkan Kit Kamera, menerapkan Lensa, dan menerbitkan umpan kamera dengan Lensa yang diterapkan ke panggung.)

JavaScript

```
<!-- Load all Desired Scripts -->
<script src="./helpers.js"></script>
<script src="./media-devices.js"></script>
<!-- <script type="module" src="./stages-simple.js"></script> -->
<script src="./dist/bundle.js"></script>
```

Menampilkan dan Mengatur Peserta

Selanjutnya, buat `helpers.js`, yang berisi metode pembantu yang akan Anda gunakan untuk menampilkan dan mengatur peserta:

JavaScript

```
/*! Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved. SPDX-License-
Identifier: Apache-2.0 */

function setupParticipant({ isLocal, id }) {
  const groupId = isLocal ? 'local-media' : 'remote-media';
  const groupContainer = document.getElementById(groupId);

  const participantContainerId = isLocal ? 'local' : id;
  const participantContainer = createContainer(participantContainerId);
  const videoEl = createVideoEl(participantContainerId);

  participantContainer.appendChild(videoEl);
  groupContainer.appendChild(participantContainer);

  return videoEl;
}

function teardownParticipant({ isLocal, id }) {
  const groupId = isLocal ? 'local-media' : 'remote-media';
  const groupContainer = document.getElementById(groupId);
  const participantContainerId = isLocal ? 'local' : id;

  const participantDiv = document.getElementById(
    participantContainerId + '-container'
  );
}
```

```
if (!participantDiv) {
  return;
}
groupContainer.removeChild(participantDiv);
}

function createVideoEl(id) {
  const videoEl = document.createElement('video');
  videoEl.id = id;
  videoEl.autoplay = true;
  videoEl.playsInline = true;
  videoEl.srcObject = new MediaStream();
  return videoEl;
}

function createContainer(id) {
  const participantContainer = document.createElement('div');
  participantContainer.classList = 'participant-container';
  participantContainer.id = id + '-container';

  return participantContainer;
}
```

Tampilkan Kamera dan Mikrofon yang Terhubung

Selanjutnya, buat `media-devices.js`, yang berisi metode pembantu untuk menampilkan kamera dan mikrofon yang terhubung ke perangkat Anda:

JavaScript

```
/*! Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved. SPDX-License-
Identifier: Apache-2.0 */

/**
 * Returns an initial list of devices populated on the page selects
 */
async function initializeDeviceSelect() {
  const videoSelectEl = document.getElementById('video-devices');
  videoSelectEl.disabled = false;

  const { videoDevices, audioDevices } = await getDevices();
  videoDevices.forEach((device, index) => {
    videoSelectEl.options[index] = new Option(device.label, device.deviceId);
  });
}
```

```
});

const audioSelectEl = document.getElementById('audio-devices');

audioSelectEl.disabled = false;
audioDevices.forEach((device, index) => {
  audioSelectEl.options[index] = new Option(device.label, device.deviceId);
});
}

/**
 * Returns all devices available on the current device
 */
async function getDevices() {
  // Prevents issues on Safari/FF so devices are not blank
  await navigator.mediaDevices.getUserMedia({ video: true, audio: true });

  const devices = await navigator.mediaDevices.enumerateDevices();
  // Get all video devices
  const videoDevices = devices.filter((d) => d.kind === 'videoinput');
  if (!videoDevices.length) {
    console.error('No video devices found.');
```

```
  }
}
```

```
// Get all audio devices
```

```
const audioDevices = devices.filter((d) => d.kind === 'audioinput');
```

```
if (!audioDevices.length) {
```

```
  console.error('No audio devices found.');
```

```
}
```

```
return { videoDevices, audioDevices };
```

```
}
```

```
async function getCamera(deviceId) {
```

```
  // Use Max Width and Height
```

```
  return navigator.mediaDevices.getUserMedia({
```

```
    video: {
```

```
      deviceId: deviceId ? { exact: deviceId } : null,
```

```
    },
```

```
    audio: false,
```

```
  });
```

```
}
```

```
async function getMic(deviceId) {
```

```
return navigator.mediaDevices.getUserMedia({
  video: false,
  audio: {
    deviceId: deviceId ? { exact: deviceId } : null,
  },
});
}
```

Buat Sesi Kit Kamera

Buat `stages.js`, yang berisi logika untuk menerapkan Lens ke umpan kamera dan mempublikasikan umpan ke panggung. Di bagian pertama file ini, kami mengimpor SDK siaran dan SDK Web Kit Kamera dan menginisialisasi variabel yang akan kami gunakan dengan setiap SDK. Kami membuat sesi Kit Kamera dengan menelepon `createSession` setelah [bootstrap SDK Web Kit Kamera](#). Perhatikan bahwa objek elemen kanvas diteruskan ke sesi; ini memberitahu Kit Kamera untuk merender ke kanvas itu.

Java

```
/*! Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved. SPDX-License-
Identifier: Apache-2.0 */

// All helpers are expose on 'media-devices.js' and 'dom.js'
// const { setupParticipant } = window;
// const { initializeDeviceSelect, getCamera, getMic } = window;
// require('./helpers.js');
// require('./media-devices.js');

const {
  Stage,
  LocalStageStream,
  SubscribeType,
  StageEvents,
  ConnectionState,
  StreamType,
} = IVSBroadcastClient;

import {
  bootstrapCameraKit,
  createMediaStreamSource,
  Transform2D,
} from '@snap/camera-kit';
```

```
let cameraButton = document.getElementById('camera-control');
let micButton = document.getElementById('mic-control');
let joinButton = document.getElementById('join-button');
let leaveButton = document.getElementById('leave-button');

let controls = document.getElementById('local-controls');
let videoDevicesList = document.getElementById('video-devices');
let audioDevicesList = document.getElementById('audio-devices');

// Stage management
let stage;
let joining = false;
let connected = false;
let localCamera;
let localMic;
let cameraStageStream;
let micStageStream;

const liveRenderTarget = document.getElementById('canvas');

const init = async () => {
  await initializeDeviceSelect();

  const cameraKit = await bootstrapCameraKit({
    apiToken: INSERT_API_TOKEN_HERE,
  });

  const session = await cameraKit.createSession({ liveRenderTarget });
```

Ambil dan Terapkan Lensa

Untuk mengambil Lensa Anda, masukkan ID Grup Lensa Anda, yang dapat ditemukan di [Portal Pengembang Kit Kamera](#). Dalam contoh ini, kita tetap sederhana dengan menerapkan Lens pertama dalam array Lens yang dikembalikan.

JavaScript

```
const { lenses } = await cameraKit.lensRepository.loadLensGroups([
  INSERT_LENS_GROUP_ID_HERE,
]);

session.applyLens(lenses[0]);
```

Render Output dari Sesi Kit Kamera ke Kanvas

Gunakan metode [captureStream](#) untuk mengembalikan `MediaStream` konten kanvas. Kanvas akan berisi aliran video dari umpan kamera dengan Lensa diterapkan. Selain itu, tambahkan pendengar acara untuk tombol untuk membisukan kamera dan mikrofon serta pendengar acara untuk bergabung dan meninggalkan panggung. Dalam acara pendengar untuk bergabung dengan panggung, kami meneruskan sesi Kit Kamera dan `MediaStream` dari kanvas sehingga dapat dipublikasikan ke panggung.

JavaScript

```
const snapStream = liveRenderTarget.captureStream();

cameraButton.addEventListener('click', () => {
  const isMuted = !cameraStageStream.isMuted;
  cameraStageStream.setMuted(isMuted);
  cameraButton.innerText = isMuted ? 'Show Camera' : 'Hide Camera';
});

micButton.addEventListener('click', () => {
  const isMuted = !micStageStream.isMuted;
  micStageStream.setMuted(isMuted);
  micButton.innerText = isMuted ? 'Unmute Mic' : 'Mute Mic';
});

joinButton.addEventListener('click', () => {
  joinStage(session, snapStream);
});

leaveButton.addEventListener('click', () => {
  leaveStage();
});
};
```

Menyediakan Kit Kamera dengan Sumber Media untuk Rendering dan Publikasikan `LocalStageStream`

Untuk mempublikasikan aliran video dengan Lensa diterapkan, buat fungsi yang dipanggil `setCameraKitSource` untuk meneruskan yang `MediaStream` diambil dari kanvas sebelumnya. `MediaStream` Dari kanvas tidak melakukan apa-apa saat ini karena kami belum memasukkan umpan kamera lokal kami. Kami dapat menggabungkan umpan kamera lokal kami dengan memanggil metode `getCamera` pembantu dan menetapkannya. `localCamera` Kami

kemudian dapat meneruskan umpan kamera lokal kami (`viaLocalCamera`) dan objek sesi `setCameraKitSource`. `setCameraKitSource` Fungsi ini mengubah umpan kamera lokal kami ke [sumber media CameraKit](#) dengan menelepon `createMediaStreamSource`. Sumber media untuk kemudian `CameraKit` [diubah](#) untuk mencerminkan kamera yang menghadap ke depan. Efek Lens kemudian diterapkan ke sumber media dan dirender ke kanvas keluaran dengan memanggil `session.play()`.

Dengan Lens sekarang diterapkan pada yang `MediaStream` ditangkap dari kanvas, kita kemudian dapat melanjutkan untuk menerbitkannya ke panggung. Kami melakukannya dengan membuat `LocalStageStream` dengan trek video dari `MediaStream`. Sebuah contoh kemudian `LocalStageStream` dapat diteruskan ke a `StageStrategy` untuk dipublikasikan.

JavaScript

```
async function setCameraKitSource(session, mediaStream) {
  const source = createMediaStreamSource(mediaStream);
  await session.setSource(source);
  source.setTransform(Transform2D.MirrorX);
  session.play();
}

const joinStage = async (session, snapStream) => {
  if (connected || joining) {
    return;
  }
  joining = true;

  const token = document.getElementById('token').value;

  if (!token) {
    window.alert('Please enter a participant token');
    joining = false;
    return;
  }

  // Retrieve the User Media currently set on the page
  localCamera = await getCamera(videoDevicesList.value);
  localMic = await getMic(audioDevicesList.value);
  await setCameraKitSource(session, localCamera);
  // Create StageStreams for Audio and Video
  // cameraStageStream = new LocalStageStream(localCamera.getVideoTracks()[0]);
  cameraStageStream = new LocalStageStream(snapStream.getVideoTracks()[0]);
}
```

```
micStageStream = new LocalStageStream(localMic.getAudioTracks()[0]);

const strategy = {
  stageStreamsToPublish() {
    return [cameraStageStream, micStageStream];
  },
  shouldPublishParticipant() {
    return true;
  },
  shouldSubscribeToParticipant() {
    return SubscribeType.AUDIO_VIDEO;
  },
};
```

Kode yang tersisa di bawah ini adalah untuk membuat dan mengelola tahap kami:

JavaScript

```
stage = new Stage(token, strategy);

// Other available events:
// https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-guides/stages#events

stage.on(StageEvents.STAGE_CONNECTION_STATE_CHANGED, (state) => {
  connected = state === ConnectionState.CONNECTED;

  if (connected) {
    joining = false;
    controls.classList.remove('hidden');
  } else {
    controls.classList.add('hidden');
  }
});

stage.on(StageEvents.STAGE_PARTICIPANT_JOINED, (participant) => {
  console.log('Participant Joined:', participant);
});

stage.on(
  StageEvents.STAGE_PARTICIPANT_STREAMS_ADDED,
  (participant, streams) => {
    console.log('Participant Media Added: ', participant, streams);

    let streamsToDisplay = streams;
```

```
    if (participant.isLocal) {
      // Ensure to exclude local audio streams, otherwise echo will occur
      streamsToDisplay = streams.filter(
        (stream) => stream.streamType === StreamType.VIDEO
      );
    }

    const videoEl = setupParticipant(participant);
    streamsToDisplay.forEach((stream) =>
      videoEl.srcObject.addTrack(stream.mediaStreamTrack)
    );
  }
);

stage.on(StageEvents.STAGE_PARTICIPANT_LEFT, (participant) => {
  console.log('Participant Left: ', participant);
  teardownParticipant(participant);
});

try {
  await stage.join();
} catch (err) {
  joining = false;
  connected = false;
  console.error(err.message);
}
};

const leaveStage = async () => {
  stage.leave();

  joining = false;
  connected = false;

  cameraButton.innerText = 'Hide Camera';
  micButton.innerText = 'Mute Mic';
  controls.classList.add('hidden');
};

init();
```

Membuat File Konfigurasi Webpack

Buat `webpack.config.js` dan tambahkan kode berikut. Ini menggabungkan logika di atas sehingga Anda dapat menggunakan pernyataan impor untuk menggunakan Kit Kamera.

JavaScript

```
const path = require('path');
module.exports = {
  entry: ['./stage.js'],
  output: {
    filename: 'bundle.js',
    path: path.resolve(__dirname, 'dist'),
  },
};
```

Terakhir, jalankan `npm run build` untuk menggabungkan Anda JavaScript seperti yang didefinisikan dalam file konfigurasi Webpack. Anda kemudian dapat melayani HTML dan JavaScript dari server web. Misalnya, Anda dapat menggunakan server HTTP Python dan membuka `localhost:8000` untuk melihat hasilnya:

```
# Run this from the command line and the directory containing index.html
python3 -m http.server -d ./
```

Android

Untuk mengintegrasikan SDK Kit Kamera Snap dengan SDK siaran Android IVS, Anda harus menginstal SDK Kit Kamera, menginisialisasi sesi Kit Kamera, menerapkan Lensa, dan memasukkan output sesi Kit Kamera ke sumber input gambar khusus.

Untuk menginstal SDK Kit Kamera, tambahkan yang berikut ini ke `build.gradle` file modul Anda. Ganti `$cameraKitVersion` dengan [versi SDK Kit Kamera terbaru](#).

Java

```
implementation "com.snap.camerakit:camerakit:$cameraKitVersion"
```

Inisialisasi dan dapatkan `a. cameraKitSession` Camera Kit juga menyediakan pembungkus yang nyaman untuk [CameraX](#) API Android, jadi Anda tidak perlu menulis logika rumit untuk menggunakan

CameraX dengan Camera Kit. Anda dapat menggunakan `CameraXImageProcessorSource` objek sebagai [Sumber](#) untuk [ImageProcessor](#), yang memungkinkan Anda untuk memulai frame streaming pratinjau kamera.

Java

```
protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    setContentView(R.layout.activity_main);

    // Camera Kit support implementation of ImageProcessor that is backed by
    CameraX library:
    // https://developer.android.com/training/camerax
    CameraXImageProcessorSource imageProcessorSource = new
    CameraXImageProcessorSource(
        this /*context*/, this /*lifecycleOwner*/
    );
    imageProcessorSource.startPreview(true /*cameraFacingFront*/);

    cameraKitSession = Sessions.newBuilder(this)
        .imageProcessorSource(imageProcessorSource)
        .attachTo(findViewById(R.id.camerakit_stub))
        .build();
}
```

Ambil dan Terapkan Lensa

Anda dapat mengonfigurasi Lensa dan urutannya di korsel di Portal [Pengembang Kit Kamera](#):

Java

```
// Fetch lenses from repository and apply them
// Replace LENS_GROUP_ID with Lens Group ID from https://camera-kit.snapchat.com
cameraKitSession.getLenses().getRepository().get(new Available(LENS_GROUP_ID),
    available -> {
        Log.d(TAG, "Available lenses: " + available);
        Lenses.whenHasFirst(available, lens ->
            cameraKitSession.getLenses().getProcessor().apply(lens, result -> {
                Log.d(TAG, "Apply lens [" + lens + "] success: " + result);
            }));
    });
});
```

Untuk menyiarkan, kirim bingkai yang diproses ke Surface dasar sumber gambar kustom. Gunakan `DeviceDiscovery` objek dan buat `CustomImageSource` untuk mengembalikan `aSurfaceSource`. Anda kemudian dapat merender output dari `CameraKit` sesi ke dasar yang `Surface` disediakan oleh `SurfaceSource`.

Java

```
val publishStreams = ArrayList<LocalStageStream>()

val deviceDiscovery = DeviceDiscovery(applicationContext)
val customSource =
    deviceDiscovery.createImageInputSource(BroadcastConfiguration.Vec2(720f, 1280f))

cameraKitSession.processor.connectOutput(outputFrom(customSource.inputSurface))
val customStream = ImageLocalStageStream(customSource)

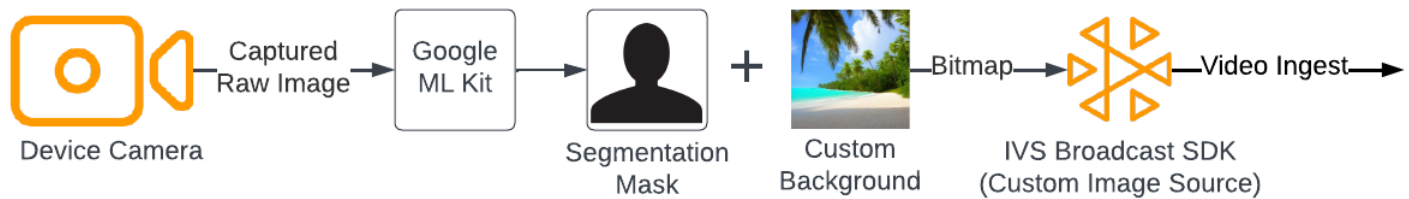
// After rendering the output from a Camera Kit session to the Surface, you can
// then return it as a LocalStageStream to be published by the Broadcast SDK
val customStream: ImageLocalStageStream = ImageLocalStageStream(surfaceSource)
publishStreams.add(customStream)

@Override
fun stageStreamsToPublishForParticipant(stage: Stage, participantInfo:
    ParticipantInfo): List<LocalStageStream> = publishStreams
```

Penggantian Latar Belakang

Penggantian latar belakang adalah jenis filter kamera yang memungkinkan pembuat live-stream mengubah latar belakang mereka. Seperti yang ditunjukkan pada diagram berikut, mengganti latar belakang Anda melibatkan:

1. Mendapatkan gambar kamera dari umpan kamera langsung.
2. Mensegmentasinya menjadi komponen latar depan dan latar belakang menggunakan Google ML Kit.
3. Menggabungkan topeng segmentasi yang dihasilkan dengan gambar latar belakang khusus.
4. Meneruskannya ke Sumber Gambar Kustom untuk disiarkan.



Web

Bagian ini mengasumsikan Anda sudah terbiasa dengan [penerbitan dan berlangganan video menggunakan Web Broadcast SDK](#).

Untuk mengganti latar belakang live stream dengan gambar kustom, gunakan [model segmentasi selfie](#) dengan [MediaPipe Image Segmenter](#). Ini adalah model pembelajaran mesin yang mengidentifikasi piksel mana dalam bingkai video yang berada di latar depan atau latar belakang. Anda kemudian dapat menggunakan hasil dari model untuk mengganti latar belakang streaming langsung, dengan menyalin piksel latar depan dari umpan video ke gambar khusus yang mewakili latar belakang baru.

Untuk mengintegrasikan penggantian latar belakang dengan SDK siaran Web streaming real-time IVS, Anda perlu:

1. Instal MediaPipe dan Webpack. (Contoh kami menggunakan Webpack sebagai bundler, tetapi Anda dapat menggunakan bundler pilihan Anda.)
2. Buat `index.html`.
3. Tambahkan elemen media.
4. Tambahkan tag skrip.
5. Buat `app.js`.
6. Muat gambar latar belakang kustom.
7. Buat instans `ImageSegmenter`.
8. Render umpan video ke kanvas.
9. Buat logika penggantian latar belakang.
10. Buat File konfigurasi Webpack.
11. Bundel JavaScript file Anda.

Instal MediaPipe dan Webpack

Untuk memulai, instal paket `@mediapipe/tasks-vision` dan `webpack` npm. Contoh di bawah ini menggunakan Webpack sebagai JavaScript bundler; Anda dapat menggunakan bundler yang berbeda jika diinginkan.

JavaScript

```
npm i @mediapipe/tasks-vision webpack webpack-cli
```

Pastikan juga memperbarui `package.json` untuk menentukan webpack skrip build Anda:

JavaScript

```
"scripts": {  
  "test": "echo \"Error: no test specified\" && exit 1",  
  "build": "webpack"  
},
```

Buat index.html

Selanjutnya, buat boilerplate HTML dan impor SDK siaran Web sebagai tag skrip. Dalam kode berikut, pastikan untuk mengganti `<SDK version>` dengan versi SDK siaran yang Anda gunakan.

JavaScript

```
<!DOCTYPE html>  
<html lang="en">  
  
<head>  
  <meta charset="UTF-8" />  
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />  
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />  
  
  <!-- Import the SDK -->  
  <script src="https://web-broadcast.live-video.net/<SDK version>/amazon-ivs-web-broadcast.js"></script>  
</head>  
  
<body>
```



```
</body>  
</html>
```

Tambahkan Elemen Media

Selanjutnya, tambahkan elemen video dan dua elemen kanvas dalam tag tubuh. Elemen video akan berisi umpan kamera langsung Anda dan akan digunakan sebagai input ke MediaPipe Segmenter Gambar. Elemen kanvas pertama akan digunakan untuk membuat pratinjau umpan yang akan disiarkan. Elemen kanvas kedua akan digunakan untuk membuat gambar kustom yang akan digunakan sebagai latar belakang. Karena kanvas kedua dengan gambar khusus hanya digunakan sebagai sumber untuk menyalin piksel secara terprogram dari itu ke kanvas akhir, itu disembunyikan dari pandangan.

JavaScript

```
<div class="row local-container">  
  <video id="webcam" autoplay style="display: none"></video>  
</div>  
<div class="row local-container">  
  <canvas id="canvas" width="640px" height="480px"></canvas>  
  
  <div class="column" id="local-media"></div>  
  <div class="static-controls hidden" id="local-controls">  
    <button class="button" id="mic-control">Mute Mic</button>  
    <button class="button" id="camera-control">Mute Camera</button>  
  </div>  
</div>  
<div class="row local-container">  
  <canvas id="background" width="640px" height="480px" style="display: none"></  
canvas>  
</div>
```

Tambahkan Tag Skrip

Tambahkan tag skrip untuk memuat JavaScript file yang dibundel yang akan berisi kode untuk melakukan penggantian latar belakang dan mempublikasikannya ke tahap:

```
<script src="./dist/bundle.js"></script>
```

Buat app.js

Selanjutnya, buat JavaScript file untuk mendapatkan objek elemen untuk kanvas dan elemen video yang dibuat di halaman HTML. Impor `FilesetResolver` modul `ImageSegmenter` dan `ImageSegmenterModul` ini akan digunakan untuk melakukan tugas segmentasi.

JavaScript

```
const canvasElement = document.getElementById("canvas");
const background = document.getElementById("background");
const canvasCtx = canvasElement.getContext("2d");
const backgroundCtx = background.getContext("2d");
const video = document.getElementById("webcam");

import { ImageSegmenter, FilesetResolver } from "@mediapipe/tasks-vision";
```

Selanjutnya, buat fungsi yang dipanggil `init()` untuk mengambil `MediaStream` dari kamera pengguna dan memanggil fungsi callback setiap kali bingkai kamera selesai dimuat. Tambahkan pendengar acara untuk tombol untuk bergabung dan meninggalkan panggung.

Perhatikan bahwa ketika bergabung dengan tahap, kita meneruskan variabel bernama `segmentationStream`. Ini adalah aliran video yang diambil dari elemen kanvas, berisi gambar latar depan yang dilapisi pada gambar khusus yang mewakili latar belakang. Nantinya, aliran kustom ini akan digunakan untuk membuat instance dari `LocalStageStream`, yang dapat dipublikasikan ke panggung.

JavaScript

```
const init = async () => {
  await initializeDeviceSelect();

  cameraButton.addEventListener("click", () => {
    const isMuted = !cameraStageStream.isMuted;
    cameraStageStream.setMuted(isMuted);
    cameraButton.innerText = isMuted ? "Show Camera" : "Hide Camera";
  });

  micButton.addEventListener("click", () => {
    const isMuted = !micStageStream.isMuted;
    micStageStream.setMuted(isMuted);
    micButton.innerText = isMuted ? "Unmute Mic" : "Mute Mic";
  });
};
```

```
localCamera = await getCamera(videoDevicesList.value);
const segmentationStream = canvasElement.captureStream();

joinButton.addEventListener("click", () => {
  joinStage(segmentationStream);
});

leaveButton.addEventListener("click", () => {
  leaveStage();
});
};
```

Muat Gambar Latar Belakang Kustom

Di bagian bawah `init` fungsi, tambahkan kode untuk memanggil fungsi `bernainitBackgroundCanvas`, yang memuat gambar kustom dari file lokal dan merendernya ke kanvas. Kami akan mendefinisikan fungsi ini pada langkah berikutnya. Tetapkan yang `MediaStream` diambil dari kamera pengguna ke objek `video`. Nantinya, objek `video` ini akan diteruskan ke `Image Segmenter`. Juga, atur fungsi bernama `renderVideoToCanvas` sebagai fungsi panggilan balik untuk dipanggil setiap kali bingkai `video` selesai dimuat. Kami akan mendefinisikan fungsi ini di langkah selanjutnya.

JavaScript

```
initBackgroundCanvas();

video.srcObject = localCamera;
video.addEventListener("loadeddata", renderVideoToCanvas);
```

Mari kita implementasikan `initBackgroundCanvas` fungsi, yang memuat gambar dari file lokal. Dalam contoh ini, kami menggunakan gambar pantai sebagai latar belakang khusus. Kanvas yang berisi gambar khusus akan disembunyikan dari tampilan, karena Anda akan menggabungkannya dengan piksel latar depan dari elemen kanvas yang berisi umpan kamera.

JavaScript

```
const initBackgroundCanvas = () => {
  let img = new Image();
  img.src = "beach.jpg";

  img.onload = () => {
```

```

    backgroundCtx.clearRect(0, 0, canvas.width, canvas.height);
    backgroundCtx.drawImage(img, 0, 0);
  };
};

```

Buat sebuah Instance dari ImageSegmenter

Selanjutnya, buat instance `ImageSegmenter`, yang akan mengelompokkan gambar dan mengembalikan hasilnya sebagai topeng. Saat membuat instance `ImageSegmenter`, Anda akan menggunakan [model segmentasi selfie](#).

JavaScript

```

const createImageSegmenter = async () => {
  const audio = await FilesetResolver.forVisionTasks("https://cdn.jsdelivr.net/npm/@mediapipe/tasks-vision@0.10.2/wasm");

  imageSegmenter = await ImageSegmenter.createFromOptions(audio, {
    baseOptions: {
      modelAssetPath: "https://storage.googleapis.com/mediapipe-models/image_segementer/selfie_segementer/float16/latest/selfie_segementer.tflite",
      delegate: "GPU",
    },
    runningMode: "VIDEO",
    outputCategoryMask: true,
  });
};

```

Render Umpan Video ke Kanvas

Selanjutnya, buat fungsi yang merender umpan video ke elemen kanvas lainnya. Kita perlu merender umpan video ke kanvas sehingga kita dapat mengekstrak piksel latar depan darinya menggunakan Canvas 2D API. Saat melakukan ini, kami juga akan meneruskan bingkai video ke instance `ImageSegmenter`, menggunakan metode [SegmentForVideo](#) untuk mengelompokkan latar depan dari latar belakang dalam bingkai video. Ketika metode [SegmentForVideo](#) kembali, ia memanggil fungsi callback kustom kami, `replaceBackground`, untuk melakukan penggantian latar belakang.

JavaScript

```

const renderVideoToCanvas = async () => {
  if (video.currentTime === lastWebcamTime) {

```

```
    window.requestAnimationFrame(renderVideoToCanvas);
    return;
}
lastWebcamTime = video.currentTime;
canvasCtx.drawImage(video, 0, 0, video.videoWidth, video.videoHeight);

if (imageSegmenter === undefined) {
    return;
}

let startTimeMs = performance.now();

imageSegmenter.segmentForVideo(video, startTimeMs, replaceBackground);
};
```

Buat Logika Penggantian Latar Belakang

Buat `replaceBackground` fungsi, yang menggabungkan gambar latar belakang kustom dengan latar depan dari umpan kamera untuk menggantikan latar belakang. Fungsi pertama mengambil data piksel yang mendasari gambar latar belakang kustom dan umpan video dari dua elemen kanvas yang dibuat sebelumnya. Kemudian iterasi melalui topeng yang disediakan oleh `ImageSegmenter`, yang menunjukkan piksel mana yang ada di latar depan. Saat beralih melalui topeng, ia secara selektif menyalin piksel yang berisi umpan kamera pengguna ke data piksel latar belakang yang sesuai. Setelah selesai, itu mengubah data piksel akhir dengan latar depan disalin ke latar belakang dan menariknya ke Canvas.

JavaScript

```
function replaceBackground(result) {
    let imageData = canvasCtx.getImageData(0, 0, video.videoWidth,
    video.videoHeight).data;
    let backgroundData = backgroundCtx.getImageData(0, 0, video.videoWidth,
    video.videoHeight).data;
    const mask = result.categoryMask.getAsFloat32Array();
    let j = 0;

    for (let i = 0; i < mask.length; ++i) {
        const maskVal = Math.round(mask[i] * 255.0);

        j += 4;
        // Only copy pixels on to the background image if the mask indicates they are in the
        foreground
    }
}
```

```

    if (maskVal < 255) {
      backgroundData[j] = imageData[j];
      backgroundData[j + 1] = imageData[j + 1];
      backgroundData[j + 2] = imageData[j + 2];
      backgroundData[j + 3] = imageData[j + 3];
    }
  }

  // Convert the pixel data to a format suitable to be drawn to a canvas
  const uint8Array = new Uint8ClampedArray(backgroundData.buffer);
  const dataNew = new ImageData(uint8Array, video.videoWidth, video.videoHeight);
  canvasCtx.putImageData(dataNew, 0, 0);
  window.requestAnimationFrame(renderVideoToCanvas);
}

```

Untuk referensi, berikut adalah `app.js` file lengkap yang berisi semua logika di atas:

JavaScript

```

/*! Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved. SPDX-License-
Identifier: Apache-2.0 */

// All helpers are expose on 'media-devices.js' and 'dom.js'
const { setupParticipant } = window;

const { Stage, LocalStageStream, SubscribeType, StageEvents, ConnectionState,
  StreamType } = IVSBroadcastClient;
const canvasElement = document.getElementById("canvas");
const background = document.getElementById("background");
const canvasCtx = canvasElement.getContext("2d");
const backgroundCtx = background.getContext("2d");
const video = document.getElementById("webcam");

import { ImageSegmenter, FilesetResolver } from "@mediapipe/tasks-vision";

let cameraButton = document.getElementById("camera-control");
let micButton = document.getElementById("mic-control");
let joinButton = document.getElementById("join-button");
let leaveButton = document.getElementById("leave-button");

let controls = document.getElementById("local-controls");
let audioDevicesList = document.getElementById("audio-devices");
let videoDevicesList = document.getElementById("video-devices");

```

```
// Stage management
let stage;
let joining = false;
let connected = false;
let localCamera;
let localMic;
let cameraStageStream;
let micStageStream;
let imageSegmenter;
let lastWebcamTime = -1;

const init = async () => {
  await initializeDeviceSelect();

  cameraButton.addEventListener("click", () => {
    const isMuted = !cameraStageStream.isMuted;
    cameraStageStream.setMuted(isMuted);
    cameraButton.innerText = isMuted ? "Show Camera" : "Hide Camera";
  });

  micButton.addEventListener("click", () => {
    const isMuted = !micStageStream.isMuted;
    micStageStream.setMuted(isMuted);
    micButton.innerText = isMuted ? "Unmute Mic" : "Mute Mic";
  });

  localCamera = await getCamera(videoDevicesList.value);
  const segmentationStream = canvasElement.captureStream();

  joinButton.addEventListener("click", () => {
    joinStage(segmentationStream);
  });

  leaveButton.addEventListener("click", () => {
    leaveStage();
  });

  initBackgroundCanvas();

  video.srcObject = localCamera;
  video.addEventListener("loadeddata", renderVideoToCanvas);
};

const joinStage = async (segmentationStream) => {
```

```
if (connected || joining) {
  return;
}
joining = true;

const token = document.getElementById("token").value;

if (!token) {
  window.alert("Please enter a participant token");
  joining = false;
  return;
}

// Retrieve the User Media currently set on the page
localMic = await getMic(audioDevicesList.value);

cameraStageStream = new LocalStageStream(segmentationStream.getVideoTracks()[0]);
micStageStream = new LocalStageStream(localMic.getAudioTracks()[0]);

const strategy = {
  stageStreamsToPublish() {
    return [cameraStageStream, micStageStream];
  },
  shouldPublishParticipant() {
    return true;
  },
  shouldSubscribeToParticipant() {
    return SubscribeType.AUDIO_VIDEO;
  },
};

stage = new Stage(token, strategy);

// Other available events:
// https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-guides/stages#events
stage.on(StageEvents.STAGE_CONNECTION_STATE_CHANGED, (state) => {
  connected = state === ConnectionState.CONNECTED;

  if (connected) {
    joining = false;
    controls.classList.remove("hidden");
  } else {
    controls.classList.add("hidden");
  }
}
```



```
});

stage.on(StageEvents.STAGE_PARTICIPANT_JOINED, (participant) => {
  console.log("Participant Joined:", participant);
});

stage.on(StageEvents.STAGE_PARTICIPANT_STREAMS_ADDED, (participant, streams) => {
  console.log("Participant Media Added: ", participant, streams);

  let streamsToDisplay = streams;

  if (participant.isLocal) {
    // Ensure to exclude local audio streams, otherwise echo will occur
    streamsToDisplay = streams.filter((stream) => stream.streamType !==
StreamType.VIDEO);
  }

  const videoEl = setupParticipant(participant);
  streamsToDisplay.forEach((stream) =>
videoEl.srcObject.addTrack(stream.mediaStreamTrack));
});

stage.on(StageEvents.STAGE_PARTICIPANT_LEFT, (participant) => {
  console.log("Participant Left: ", participant);
  teardownParticipant(participant);
});

try {
  await stage.join();
} catch (err) {
  joining = false;
  connected = false;
  console.error(err.message);
}
};

const leaveStage = async () => {
  stage.leave();

  joining = false;
  connected = false;

  cameraButton.innerText = "Hide Camera";
  micButton.innerText = "Mute Mic";
```

```
controls.classList.add("hidden");
};

function replaceBackground(result) {
  let imageData = canvasCtx.getImageData(0, 0, video.videoWidth,
video.videoHeight).data;
  let backgroundData = backgroundCtx.getImageData(0, 0, video.videoWidth,
video.videoHeight).data;
  const mask = result.categoryMask.getAsFloat32Array();
  let j = 0;

  for (let i = 0; i < mask.length; ++i) {
    const maskVal = Math.round(mask[i] * 255.0);

    j += 4;
    if (maskVal < 255) {
      backgroundData[j] = imageData[j];
      backgroundData[j + 1] = imageData[j + 1];
      backgroundData[j + 2] = imageData[j + 2];
      backgroundData[j + 3] = imageData[j + 3];
    }
  }
  const uint8Array = new Uint8ClampedArray(backgroundData.buffer);
  const dataNew = new ImageData(uint8Array, video.videoWidth, video.videoHeight);
  canvasCtx.putImageData(dataNew, 0, 0);
  window.requestAnimationFrame(renderVideoToCanvas);
}

const createImageSegmenter = async () => {
  const audio = await FilesetResolver.forVisionTasks("https://cdn.jsdelivr.net/npm/
@mediapipe/tasks-vision@0.10.2/wasm");

  imageSegmenter = await ImageSegmenter.createFromOptions(audio, {
    baseOptions: {
      modelAssetPath: "https://storage.googleapis.com/mediapipe-models/image_segementer/
selfie_segementer/float16/latest/selfie_segementer.tflite",
      delegate: "GPU",
    },
    runningMode: "VIDEO",
    outputCategoryMask: true,
  });
};

const renderVideoToCanvas = async () => {
```

```
if (video.currentTime === lastWebcamTime) {
  window.requestAnimationFrame(renderVideoToCanvas);
  return;
}
lastWebcamTime = video.currentTime;
canvasCtx.drawImage(video, 0, 0, video.videoWidth, video.videoHeight);

if (imageSegmenter === undefined) {
  return;
}

let startTimeMs = performance.now();

imageSegmenter.segmentForVideo(video, startTimeMs, replaceBackground);
};

const initBackgroundCanvas = () => {
  let img = new Image();
  img.src = "beach.jpg";

  img.onload = () => {
    backgroundCtx.clearRect(0, 0, canvas.width, canvas.height);
    backgroundCtx.drawImage(img, 0, 0);
  };
};

createImageSegmenter();
init();
```

Membuat File Konfigurasi Webpack

Tambahkan konfigurasi ini ke file konfigurasi Webpack Anda ke `bundelapp.js`, sehingga panggilan `import` akan berfungsi:

JavaScript

```
const path = require("path");
module.exports = {
  entry: ["/app.js"],
  output: {
    filename: "bundle.js",
    path: path.resolve(__dirname, "dist"),
  },
};
```

```
};
```

Bundel JavaScript file Anda

```
npm run build
```

Mulai server HTTP sederhana dari direktori yang berisi `index.html` dan buka `localhost:8000` untuk melihat hasilnya:

```
python3 -m http.server -d ./
```

Android

Untuk mengganti latar belakang di live stream Anda, Anda dapat menggunakan API segmentasi selfie dari [Google ML Kit](#). API segmentasi selfie menerima gambar kamera sebagai input dan mengembalikan topeng yang memberikan skor kepercayaan untuk setiap piksel gambar, yang menunjukkan apakah itu di latar depan atau latar belakang. Berdasarkan skor kepercayaan, Anda kemudian dapat mengambil warna piksel yang sesuai dari gambar latar belakang atau gambar latar depan. Proses ini berlanjut sampai semua skor kepercayaan pada topeng telah diperiksa. Hasilnya adalah array baru warna piksel yang berisi piksel latar depan dikombinasikan dengan piksel dari gambar latar belakang.

Untuk mengintegrasikan penggantian latar belakang dengan SDK siaran Android streaming real-time IVS, Anda perlu:

1. Instal pustaka CameraX dan kit Google ML.
2. Inisialisasi variabel boilerplate.
3. Buat sumber gambar khusus.
4. Kelola bingkai kamera.
5. Lewatkan bingkai kamera ke Google ML Kit.
6. Hamparkan latar depan bingkai kamera ke latar belakang kustom Anda.
7. Umpan gambar baru ke sumber gambar khusus.

Instal CameraX Libraries dan Google ML Kit

Untuk mengekstrak gambar dari umpan kamera langsung, gunakan pustaka CameraX Android. Untuk menginstal library CameraX dan Google ML Kit, tambahkan yang berikut ini ke file modul

Anda. build.gradle Ganti `${camerax_version}` dan `${google_ml_kit_version}` dengan versi terbaru dari pustaka [CameraX](#) dan [Google ML Kit](#), masing-masing.

Java

```
implementation "com.google.mlkit:segmentation-selfie:${google_ml_kit_version}"
implementation "androidx.camera:camera-core:${camerax_version}"
implementation "androidx.camera:camera-lifecycle:${camerax_version}"
```

Impor pustaka berikut:

Java

```
import androidx.camera.core.CameraSelector
import androidx.camera.core.ImageAnalysis
import androidx.camera.core.ImageProxy
import androidx.camera.lifecycle.ProcessCameraProvider
import com.google.mlkit.vision.segmentation.selfie.SelfieSegmenterOptions
```

Inisialisasi Variabel Boilerplate

Menginisialisasi instance dari `ImageAnalysis` dan instance `ExecutorService` dari:

Java

```
private lateinit var binding: ActivityMainBinding
private lateinit var cameraExecutor: ExecutorService
private var analysisUseCase: ImageAnalysis? = null
```

Inisialisasi instance Segmenter di `STREAM_MODE`:

Java

```
private val options =
    SelfieSegmenterOptions.Builder()
        .setDetectorMode(SelfieSegmenterOptions.STREAM_MODE)
        .build()

private val segmenter = Segmentation.getClient(options)
```

Buat Sumber Gambar Kustom

Dalam `onCreate` metode aktivitas Anda, buat instance `DeviceDiscovery` objek dan buat sumber gambar kustom. Yang `Surface` disediakan oleh `Custom Image Source` akan menerima gambar akhir, dengan latar depan dilapisi pada gambar latar belakang kustom. Anda kemudian akan membuat instance dari `ImageLocalStageStream` menggunakan `Custom Image Source`. Instance dari `ImageLocalStageStream` (dinamai `filterStream` dalam contoh ini) kemudian dapat dipublikasikan ke sebuah panggung. Lihat Panduan [SDK Broadcast Android IVS untuk petunjuk](#) tentang menyiapkan panggung. Terakhir, buat juga utas yang akan digunakan untuk mengelola kamera.

Java

```
var deviceDiscovery = DeviceDiscovery(applicationContext)
var customSource = deviceDiscovery.createImageInputSource( BroadcastConfiguration.Vec2(
    720F, 1280F
))
var surface: Surface = customSource.inputSurface
var filterStream = ImageLocalStageStream(customSource)

cameraExecutor = Executors.newSingleThreadExecutor()
```

Kelola Bingkai Kamera

Selanjutnya, buat fungsi untuk menginisialisasi kamera. Fungsi ini menggunakan perpustakaan `CameraX` untuk mengekstrak gambar dari umpan kamera langsung. Pertama, Anda membuat instance dari yang `ProcessCameraProvider` dipanggil `cameraProviderFuture`. Objek ini merupakan hasil future dari mendapatkan penyedia kamera. Kemudian Anda memuat gambar dari proyek Anda sebagai bitmap. Contoh ini menggunakan gambar pantai sebagai latar belakang, tetapi bisa berupa gambar apa pun yang Anda inginkan.

Anda kemudian menambahkan pendengar ke `cameraProviderFuture`. Pendengar ini diberitahu ketika kamera tersedia atau jika terjadi kesalahan selama proses mendapatkan penyedia kamera.

Java

```
private fun startCamera(surface: Surface) {
    val cameraProviderFuture = ProcessCameraProvider.getInstance(this)
    val imageResource = R.drawable.beach
    val bgBitmap: Bitmap = BitmapFactory.decodeResource(resources, imageResource)
    var resultBitmap: Bitmap;
```

```
cameraProviderFuture.addListener({
    val cameraProvider: ProcessCameraProvider = cameraProviderFuture.get()

    if (mediaImage != null) {
        val inputImage =
            InputImage.fromMediaImage(mediaImage,
imageProxy.imageInfo.rotationDegrees)

                resultBitmap = overlayForeground(mask, maskWidth,
maskHeight, inputBitmap, backgroundPixels)
                canvas = surface.lockCanvas(null);
                canvas.drawBitmap(resultBitmap, 0f, 0f, null)

                surface.unlockCanvasAndPost(canvas);

            }
            .addOnFailureListener { exception ->
                Log.d("App", exception.message!!)
            }
            .addOnCompleteListener {
                imageProxy.close()
            }
        }
    };

    val cameraSelector = CameraSelector.DEFAULT_FRONT_CAMERA

    try {
        // Unbind use cases before rebinding
        cameraProvider.unbindAll()

        // Bind use cases to camera
        cameraProvider.bindToLifecycle(this, cameraSelector, analysisUseCase)

    } catch (exc: Exception) {
        Log.e(TAG, "Use case binding failed", exc)
    }

}, ContextCompat.getMainExecutor(this))
```

```
}

```

Di dalam pendengar, buat `ImageAnalysis.Builder` untuk mengakses setiap bingkai individu dari umpan kamera langsung. Atur strategi tekanan balik ke `STRATEGY_KEEP_ONLY_LATEST`. Ini menjamin bahwa hanya satu bingkai kamera pada satu waktu yang dikirimkan untuk diproses. Konversikan setiap bingkai kamera individual ke bitmap, sehingga Anda dapat mengekstrak pikselnya untuk kemudian menggabungkannya dengan gambar latar belakang khusus.

Java

```
val imageAnalyzer = ImageAnalysis.Builder()
analysisUseCase = imageAnalyzer
    .setTargetResolution(Size(360, 640))
    .setBackpressureStrategy(ImageAnalysis.STRATEGY_KEEP_ONLY_LATEST)
    .build()

analysisUseCase?.setAnalyzer(cameraExecutor) { imageProxy: ImageProxy ->
    val mediaImage = imageProxy.image
    val tempBitmap = imageProxy.toBitmap();
    val inputBitmap = tempBitmap.rotate(imageProxy.imageInfo.rotationDegrees.toFloat())

```

Lewati Bingkai Kamera ke Google ML Kit

Selanjutnya, buat `InputImage` dan teruskan ke instance `Segmenter` untuk diproses. An `InputImage` dapat dibuat dari yang `ImageProxy` disediakan oleh instance `ImageAnalysis`. Setelah `InputImage` diberikan ke `Segmenter`, ia mengembalikan topeng dengan skor kepercayaan yang menunjukkan kemungkinan piksel berada di latar depan atau latar belakang. Topeng ini juga menyediakan properti lebar dan tinggi, yang akan Anda gunakan untuk membuat array baru yang berisi piksel latar belakang dari gambar latar belakang kustom yang dimuat sebelumnya.

Java

```
if (mediaImage != null) {
    val inputImage =
        InputImage.fromMediaImag

segmenter.process(inputImage)
    .addOnSuccessListener { segmentationMask ->
        val mask = segmentationMask.buffer
        val maskWidth = segmentationMask.width
        val maskHeight = segmentationMask.height

```



```
val backgroundPixels = IntArray(maskWidth * maskHeight)
bgBitmap.getPixels(backgroundPixels, 0, maskWidth, 0, 0, maskWidth, maskHeight)
```

Hamparkan Latar Depan Bingkai Kamera ke Latar Belakang Kustom Anda

Dengan topeng yang berisi skor kepercayaan, bingkai kamera sebagai bitmap, dan piksel warna dari gambar latar belakang kustom, Anda memiliki semua yang Anda butuhkan untuk melapisi latar depan ke latar belakang kustom Anda. `overlayForeground` fungsi ini kemudian dipanggil dengan parameter berikut:

Java

```
resultBitmap = overlayForeground(mask, maskWidth, maskHeight, inputBitmap,
    backgroundPixels)
```

Fungsi ini berulang melalui topeng dan memeriksa nilai kepercayaan untuk menentukan apakah akan mendapatkan warna piksel yang sesuai dari gambar latar belakang atau bingkai kamera. Jika nilai kepercayaan menunjukkan bahwa piksel di topeng kemungkinan besar berada di latar belakang, itu akan mendapatkan warna piksel yang sesuai dari gambar latar belakang; jika tidak, itu akan mendapatkan warna piksel yang sesuai dari bingkai kamera untuk membangun latar depan. Setelah fungsi selesai iterasi melalui mask, bitmap baru dibuat menggunakan array piksel warna baru dan dikembalikan. Bitmap baru ini berisi latar depan yang dilapisi pada latar belakang kustom.

Java

```
private fun overlayForeground(
    byteBuffer: ByteBuffer,
    maskWidth: Int,
    maskHeight: Int,
    cameraBitmap: Bitmap,
    backgroundPixels: IntArray
): Bitmap {
    @ColorInt val colors = IntArray(maskWidth * maskHeight)
    val cameraPixels = IntArray(maskWidth * maskHeight)

    cameraBitmap.getPixels(cameraPixels, 0, maskWidth, 0, 0, maskWidth, maskHeight)

    for (i in 0 until maskWidth * maskHeight) {
        val backgroundLikelihood: Float = 1 - byteBuffer.getFloat()

        // Apply the virtual background to the color if it's not part of the
        foreground
```

```

        if (backgroundLikelihood > 0.9) {
            // Get the corresponding pixel color from the background image
            // Set the color in the mask based on the background image pixel color
            colors[i] = backgroundPixels.get(i)
        } else {
            // Get the corresponding pixel color from the camera frame
            // Set the color in the mask based on the camera image pixel color
            colors[i] = cameraPixels.get(i)
        }
    }

    return Bitmap.createBitmap(
        colors, maskWidth, maskHeight, Bitmap.Config.ARGB_8888
    )
}

```

Umpan Gambar Baru ke Sumber Gambar Kustom

Anda kemudian dapat menulis bitmap baru ke yang Surface disediakan oleh sumber gambar khusus. Ini akan menyiarkannya ke panggung Anda.

Java

```

resultBitmap = overlayForeground(mask, inputBitmap, mutableBitmap, bgBitmap)
canvas = surface.lockCanvas(null);
canvas.drawBitmap(resultBitmap, 0f, 0f, null)

```

Berikut adalah fungsi lengkap untuk mendapatkan bingkai kamera, meneruskannya ke Segmenter, dan melapisinya di latar belakang:

Java

```

@androidx.annotation.OptIn(androidx.camera.core.ExperimentalGetImage::class)
private fun startCamera(surface: Surface) {
    val cameraProviderFuture = ProcessCameraProvider.getInstance(this)
    val imageResource = R.drawable.clouds
    val bgBitmap: Bitmap = BitmapFactory.decodeResource(resources, imageResource)
    var resultBitmap: Bitmap;

    cameraProviderFuture.addListener({
        // Used to bind the lifecycle of cameras to the lifecycle owner
        val cameraProvider: ProcessCameraProvider = cameraProviderFuture.get()
    })
}

```

```

val imageAnalyzer = ImageAnalysis.Builder()
analysisUseCase = imageAnalyzer
    .setTargetResolution(Size(720, 1280))
    .setBackpressureStrategy(ImageAnalysis.STRATEGY_KEEP_ONLY_LATEST)
    .build()

analysisUseCase!!.setAnalyzer(cameraExecutor) { imageProxy: ImageProxy ->
    val mediaImage = imageProxy.image
    val tempBitmap = imageProxy.toBitmap();
    val inputBitmap =
tempBitmap.rotate(imageProxy.imageInfo.rotationDegrees.toFloat())

    if (mediaImage != null) {
        val inputImage =
            InputImage.fromMediaImage(mediaImage,
imageProxy.imageInfo.rotationDegrees)

        segmenter.process(inputImage)
            .addOnSuccessListener { segmentationMask ->
                val mask = segmentationMask.buffer
                val maskWidth = segmentationMask.width
                val maskHeight = segmentationMask.height
                val backgroundPixels = IntArray(maskWidth * maskHeight)
                bgBitmap.getPixels(backgroundPixels, 0, maskWidth, 0, 0,
maskWidth, maskHeight)

                resultBitmap = overlayForeground(mask, maskWidth,
maskHeight, inputBitmap, backgroundPixels)
                canvas = surface.lockCanvas(null);
                canvas.drawBitmap(resultBitmap, 0f, 0f, null)

                surface.unlockCanvasAndPost(canvas);

            }
            .addOnFailureListener { exception ->
                Log.d("App", exception.message!!)
            }
            .addOnCompleteListener {
                imageProxy.close()
            }
    }
};

```

```
val cameraSelector = CameraSelector.DEFAULT_FRONT_CAMERA

try {
    // Unbind use cases before rebinding
    cameraProvider.unbindAll()

    // Bind use cases to camera
    cameraProvider.bindToLifecycle(this, cameraSelector, analysisUseCase)
} catch(exc: Exception) {
    Log.e(TAG, "Use case binding failed", exc)
}

}, ContextCompat.getMainExecutor(this))
}
```

SDK Siaran IVS: Mode Audio Seluler (Streaming Waktu Nyata)

Kualitas audio adalah bagian penting dari pengalaman media tim nyata apa pun, dan tidak ada konfigurasi one-size-fits-all audio yang paling sesuai untuk setiap kasus penggunaan. Untuk memastikan bahwa pengguna Anda memiliki pengalaman terbaik saat mendengarkan streaming real-time IVS, SDK seluler kami menyediakan beberapa konfigurasi audio preset, serta penyesuaian yang lebih kuat sesuai kebutuhan.

Pengantar

SDK siaran seluler IVS menyediakan kelas `StageAudioManager`. Kelas ini dirancang untuk menjadi titik kontak tunggal untuk mengontrol mode audio yang mendasarinya pada kedua platform. Di Android, ini mengontrol [AudioManager](#), termasuk mode audio, sumber audio, jenis konten, penggunaan, dan perangkat komunikasi. Di iOS, ia mengontrol aplikasi [AV AudioSession](#), serta apakah [VoiceProcessing](#) diaktifkan.

Penting: Jangan berinteraksi dengan `AVAudioSession` atau `AudioManager` langsung saat SDK siaran real-time IVS aktif. Melakukan hal itu dapat mengakibatkan hilangnya audio, atau audio yang direkam dari atau diputar kembali pada perangkat yang salah.

Sebelum Anda membuat Stage objek `DeviceDiscovery` atau pertama Anda, `StageAudioManager` kelas harus dikonfigurasi.

Android (Kotlin)

```
StageAudioManager.getInstance(context).setPreset(StageAudioManager.UseCasePreset.VIDEO_CHAT)
// The default value

val deviceDiscovery = DeviceDiscovery(context)
val stage = Stage(context, token, this)

// Other Stage implementation code
```

iOS (Swift)

```
IVSStageAudioManager.sharedInstance().setPreset(.videoChat) // The default value

let deviceDiscovery = IVSDeviceDiscovery()
let stage = try? IVSStage(token: token, strategy: self)

// Other Stage implementation code
```

Jika tidak ada yang diatur pada inisialisasi `StageAudioManager` sebelum `DeviceDiscovery` atau `Stage` instance, `VideoChat` preset diterapkan secara otomatis.

Preset Mode Audio

SDK siaran real-time menyediakan tiga preset, masing-masing disesuaikan dengan kasus penggunaan umum, seperti yang dijelaskan di bawah ini. Untuk setiap preset, kami mencakup lima kategori utama yang membedakan preset satu sama lain.

Obrolan Video

Ini adalah preset default, yang dirancang untuk saat perangkat lokal akan melakukan percakapan real-time dengan peserta lain.

Kategori	Android	iOS
Pembatalan Echo	Aktif	Aktif
Volume Rocker	Volume Panggilan	Volume Panggilan

Kategori	Android	iOS
Pemilihan Mikrofon	Terbatas berdasarkan OS. Mikrofon USB mungkin tidak tersedia.	Terbatas berdasarkan OS. Mikrofon USB dan Bluetooth mungkin tidak tersedia. Headset Bluetooth yang menangani input dan output bersama-sama harus berfungsi; misalnya, AirPods.
Keluaran Audio	Perangkat output apa pun harus berfungsi.	Terbatas berdasarkan OS. Headset kabel mungkin tidak tersedia.
Kualitas Audio	Sedang/Rendah. Ini akan terdengar seperti panggilan telepon, tidak seperti pemutaran media.	Sedang/Rendah. Ini akan terdengar seperti panggilan telepon, tidak seperti pemutaran media.

Hanya Berlangganan

Preset ini dirancang untuk saat Anda berencana untuk berlangganan peserta penerbitan lain tetapi tidak mempublikasikan sendiri. Ini berfokus pada kualitas audio dan mendukung semua perangkat output yang tersedia.

Kategori	Android	iOS
Pembatalan Echo	Nonaktif	Nonaktif
Volume Rocker	Volume Media	Volume Media
Pemilihan Mikrofon	N/A, preset ini tidak dirancang untuk penerbitan.	N/A, preset ini tidak dirancang untuk penerbitan.
Keluaran Audio	Perangkat output apa pun harus berfungsi.	Perangkat output apa pun harus berfungsi.

Kategori	Android	iOS
Kualitas Audio	Tinggi. Setiap jenis media harus datang dengan jelas, termasuk musik.	Tinggi. Setiap jenis media harus datang dengan jelas, termasuk musik.

Studio

Preset ini dirancang untuk berlangganan berkualitas tinggi sambil mempertahankan kemampuan untuk mempublikasikan. Ini membutuhkan perangkat keras perekaman dan pemutaran untuk memberikan pembatalan gema. Kasus penggunaan di sini adalah menggunakan mikrofon USB dan headset kabel. SDK akan mempertahankan audio berkualitas tinggi sambil mengandalkan pemisahan fisik perangkat tersebut agar tidak menyebabkan gema.

Kategori	Android	iOS
Pembatalan Echo	Nonaktif	Nonaktif
Volume Rocker	Volume Media dalam banyak kasus. Volume Panggilan saat mikrofon Bluetooth terhubung.	Volume Media
Pemilihan Mikrofon	Mikrofon apa pun harus berfungsi.	Mikrofon apa pun harus berfungsi.
Keluaran Audio	Perangkat output apa pun harus berfungsi.	Perangkat output apa pun harus berfungsi.
Kualitas Audio	<p>Tinggi. Kedua belah pihak harus dapat mengirim musik dan mendengarnya dengan jelas di sisi lain.</p> <p>Saat headset Bluetooth terhubung, kualitas audio akan turun karena mode Bluetooth SCO diaktifkan.</p>	<p>Tinggi. Kedua belah pihak harus dapat mengirim musik dan mendengarnya dengan jelas di sisi lain.</p> <p>Saat headset Bluetooth terhubung, kualitas audio mungkin turun karena mode Bluetooth SCO diaktifkan, tergantung pada headset.</p>

Kasus Penggunaan Tingkat Lanjut

Di luar preset, baik SDK siaran streaming real-time iOS dan Android memungkinkan konfigurasi mode audio platform yang mendasarinya:

- Di Android, atur [AudioSource](#), [Usage](#), dan [ContentType](#).
- Di iOS, gunakan [AV AudioSession .Category](#), [AVAudioSession. CategoryOptions](#), [AV AudioSession .Mode](#), dan kemampuan untuk beralih jika [pemrosesan suara](#) diaktifkan atau tidak saat penerbitan.

Android (Kotlin)

```
// This would act similar to the Subscribe Only preset, but it uses a different
// ContentType.
StageAudioManager.getInstance(context)
    .setConfiguration(StageAudioManager.Source.GENERIC,
                     StageAudioManager.ContentType.MOVIE,
                     StageAudioManager.Usage.MEDIA);

val stage = Stage(context, token, this)

// Other Stage implementation code
```

iOS (Swift)

```
// This would act similar to the Subscribe Only preset, but it uses a different mode
// and options.
IVSStageAudioManager.sharedInstance()
    .setCategory(.playback,
               options: [.duckOthers, .mixWithOthers],
               mode: .default)

let stage = try? IVSStage(token: token, strategy: self)

// Other Stage implementation code
```


Menerbitkan dengan Bluetooth di Android

SDK secara otomatis kembali ke VIDEO_CHAT preset di Android jika kondisi berikut terpenuhi:

- Konfigurasi yang ditetapkan tidak menggunakan nilai VOICE_COMMUNICATION penggunaan.
- Mikrofon Bluetooth terhubung ke perangkat.
- Peserta lokal menerbitkan ke Panggung.

Ini adalah batasan sistem operasi Android dalam hal bagaimana headset Bluetooth digunakan untuk merekam audio.

Integrasi dengan SDK Lain

Karena iOS dan Android hanya mendukung satu mode audio aktif per aplikasi, biasanya terjadi konflik jika aplikasi Anda menggunakan beberapa SDK yang memerlukan kontrol mode audio. Ketika Anda mengalami konflik ini, ada beberapa strategi resolusi umum untuk dicoba, dijelaskan di bawah ini.

Cocokkan Nilai Mode Audio

Dengan menggunakan opsi konfigurasi audio lanjutan IVS SDK atau fungsionalitas SDK lainnya, buat kedua SDK sejajar dengan nilai yang mendasarinya.

Agora

iOS

Di iOS, memberi tahu Agora SDK untuk tetap AVAudioSession aktif akan mencegahnya menonaktifkan saat SDK siaran streaming real-time IVS menggunakannya.

```
myRtcEngine.SetParameters("{\"che.audio.keep.audiosession\":true}");
```

Android

Hindari `setEnabledSpeakerphone` menelepon `RtcEngine`, dan menelepon `enableLocalAudio(false)` saat menerbitkan dengan SDK siaran streaming real-time IVS. Anda dapat menelepon `enableLocalAudio(true)` lagi saat IVS SDK tidak dipublikasikan.

Menggunakan Amazon EventBridge dengan IVS Real-Time Streaming

Anda dapat menggunakan Amazon EventBridge untuk memantau aliran Amazon Interactive Video Service (IVS).

Amazon IVS mengirimkan peristiwa perubahan tentang status aliran Anda ke Amazon EventBridge. Semua acara yang dikirimkan valid. Namun, acara dikirim dengan upaya terbaik, yang berarti tidak ada jaminan bahwa:

- Acara dikirimkan - Acara yang ditunjuk dapat terjadi (misalnya, peserta dipublikasikan) tetapi ada kemungkinan bahwa Amazon IVS tidak akan mengirim acara yang sesuai ke EventBridge. Amazon IVS mencoba menyampaikan acara selama beberapa jam sebelum menyerah.
- Acara yang dikirimkan akan tiba dalam jangka waktu tertentu — Anda mungkin menerima acara hingga beberapa jam.
- Acara dikirimkan secara berurutan - Acara mungkin rusak, terutama jika dikirim dalam waktu singkat satu sama lain. Misalnya, Anda dapat melihat Peserta Tidak Diterbitkan sebelum Peserta Diterbitkan.

Meskipun jarang terjadi peristiwa yang hilang, terlambat, atau tidak berurutan, Anda harus menangani kemungkinan ini jika Anda menulis program penting bisnis yang bergantung pada urutan atau keberadaan acara pemberitahuan.

Anda dapat membuat EventBridge aturan untuk salah satu acara berikut.

Jenis Acara	Peristiwa	Dikirim Ketika...
Perubahan Status Komposisi IVS	Kegagalan Tujuan	Upaya untuk output ke Destinasi gagal. Misalnya, penyiaran ke saluran gagal karena tidak ada kunci aliran atau siaran lain yang terjadi.
Perubahan Status Komposisi IVS	Tujuan Mulai	Output ke Destination berhasil dimulai.
Perubahan Status Komposisi IVS	Tujuan Akhir	Output ke Tujuan selesai.

Jenis Acara	Peristiwa	Dikirim Ketika...
Perubahan Status Komposisi IVS	Tujuan Menghubungkan Kembali	Output ke Destination terputus dan penyambungan kembali sedang dicoba.
Perubahan Status Komposisi IVS	Sesi Mulai	Sesi Komposisi telah dibuat. Peristiwa ini diaktifkan ketika pipeline proses Komposisi berhasil diinisialisasi. Pada titik ini, pipeline Komposisi telah berhasil berlangganan Stage dan menerima media dan mampu membuat video.
Perubahan Status Komposisi IVS	Akhir Sesi	Sesi Komposisi selesai.
Perubahan Status Komposisi IVS	Kegagalan Sesi	Pipa Komposisi gagal diinisialisasi karena sumber daya Stage tidak tersedia, atau kesalahan internal lainnya.
Pembaruan Tahap IVS	Peserta Diterbitkan	Seorang peserta mulai menerbitkan ke panggung.
Pembaruan Tahap IVS	Peserta Tidak Diterbitkan	Seorang peserta telah berhenti menerbitkan ke panggung.

Membuat EventBridge Aturan Amazon untuk Amazon IVS

Anda dapat membuat aturan yang memicu peristiwa yang dipancarkan oleh Amazon IVS. Ikuti langkah-langkah di [Buat aturan di Amazon EventBridge](#) di Panduan EventBridge Pengguna Amazon. Saat memilih layanan, pilih Layanan Video Interaktif (IVS).

Contoh: Perubahan Status Komposisi

Kegagalan Tujuan: Peristiwa ini dikirim ketika upaya untuk output ke Tujuan gagal. Misalnya, penyiaran ke saluran gagal karena tidak ada kunci aliran atau siaran lain yang terjadi.

```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-012345678901",
  "detail-type": "IVS Composition State Change",
  "source": "aws.ivs",
  "account": "aws_account_id",
  "time": "2017-06-12T10:23:43Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:ivs:us-east-1:aws_account_id:composition/123456789012"
  ],
  "detail": {
    "event_name": "Destination Failure",
    "stage_arn": "<stage-arn>",
    "id": "<Destination-id>",
    "reason": "eg. stream key invalid"
  }
}
```

Tujuan Mulai: Acara ini dikirim ketika output ke Tujuan berhasil dimulai.

```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-012345678901",
  "detail-type": "IVS Composition State Change",
  "source": "aws.ivs",
  "account": "aws_account_id",
  "time": "2017-06-12T10:23:43Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:ivs:us-east-1:aws_account_id:composition/123456789012"
  ],
  "detail": {
    "event_name": "Destination Start",
    "stage_arn": "<stage-arn>",
    "id": "<destination-id>",
  }
}
```

Tujuan Akhir: Acara ini dikirim ketika output ke Tujuan selesai.

```
{
```

```

"version": "0",
"id": "01234567-0123-0123-0123-012345678901",
"detail-type": "IVS Composition State Change",
"source": "aws.ivs",
"account": "aws_account_id",
"time": "2017-06-12T10:23:43Z",
"region": "us-east-1",
"resources": [
  "arn:aws:ivs:us-east-1:aws_account_id:composition/123456789012"
],
"detail": {
  "event_name": "Destination End",
  "stage_arn": "<stage-arn>",
  "id": "<Destination-id>",
}
}

```

Destination Reconnecting: Peristiwa ini dikirim ketika output ke Destination terputus dan penyambungan kembali sedang dicoba.

```

{
  "version": "0",
  "id": "01234567-0123-0123-0123-012345678901",
  "detail-type": "IVS Composition State Change",
  "source": "aws.ivs",
  "account": "aws_account_id",
  "time": "2017-06-12T10:23:43Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:ivs:us-east-1:aws_account_id:composition/123456789012"
  ],
  "detail": {
    "event_name": "Destination Reconnecting",
    "stage_arn": "<stage-arn>",
    "id": "<Destination-id>",
  }
}

```

Sesi Mulai: Acara ini dikirim ketika sesi Komposisi dibuat. Peristiwa ini diaktifkan ketika pipeline proses Komposisi berhasil diinisialisasi. Pada titik ini, pipeline Komposisi telah berhasil berlangganan Stage dan menerima media dan mampu membuat video.

```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-012345678901",
  "detail-type": "IVS Composition State Change",
  "source": "aws.ivs",
  "account": "aws_account_id",
  "time": "2017-06-12T10:23:43Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:ivs:us-east-1:aws_account_id:composition/123456789012"
  ],
  "detail": {
    "event_name": "Session Start",
    "stage_arn": "<stage-arn>"
  }
}
```

Session End: Acara ini dikirim ketika sesi Komposisi selesai dan semua sumber daya dihapus.

```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-012345678901",
  "detail-type": "IVS Composition State Change",
  "source": "aws.ivs",
  "account": "aws_account_id",
  "time": "2017-06-12T10:23:43Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:ivs:us-east-1:aws_account_id:composition/123456789012"
  ],
  "detail": {
    "event_name": "Session End",
    "stage_arn": "<stage-arn>"
  }
}
```

Kegagalan Sesi: Acara ini dikirim ketika pipeline Komposisi gagal diinisialisasi karena sumber daya Tahap tidak tersedia, tidak ada peserta yang berada di panggung, atau kesalahan internal lainnya.

```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-012345678901",
```

```
"detail-type": "IVS Composition State Change",
"source": "aws.ivs",
"account": "aws_account_id",
"time": "2017-06-12T10:23:43Z",
"region": "us-east-1",
"resources": [
  "arn:aws:ivs:us-east-1:aws_account_id:composition/123456789012"
],
"detail": {
  "event_name": "Session Failure",
  "stage_arn": "<stage-arn>",
  "reason": "eg. no participants in the stage"
}
}
```

Contoh: Pembaruan Tahap

Acara pembaruan panggung mencakup nama acara (yang mengklasifikasikan acara) dan metadata tentang acara tersebut. Metadata mencakup ID peserta yang memicu peristiwa, ID tahap dan sesi terkait, dan ID pengguna.

Peserta Diterbitkan: Acara ini dikirim ketika peserta mulai menerbitkan ke panggung.

```
{
  "version": "0",
  "id": "12345678-1a23-4567-a1bc-1a2b34567890",
  "detail-type": "IVS Stage Update",
  "source": "aws.ivs",
  "account": "123456789012",
  "time": "2020-06-23T20:12:36Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:ivs:us-west-2:123456789012:stage/AbCdef1G2hij"
  ],
  "detail": {
    "session_id": "st-1234567890",
    "event_name": "Participant Published",
    "user_id": "Your User Id",
    "participant_id": "xYz1c2d3e4f"
  }
}
```

Peserta Tidak Diterbitkan: Acara ini dikirim ketika peserta telah berhenti menerbitkan ke panggung.

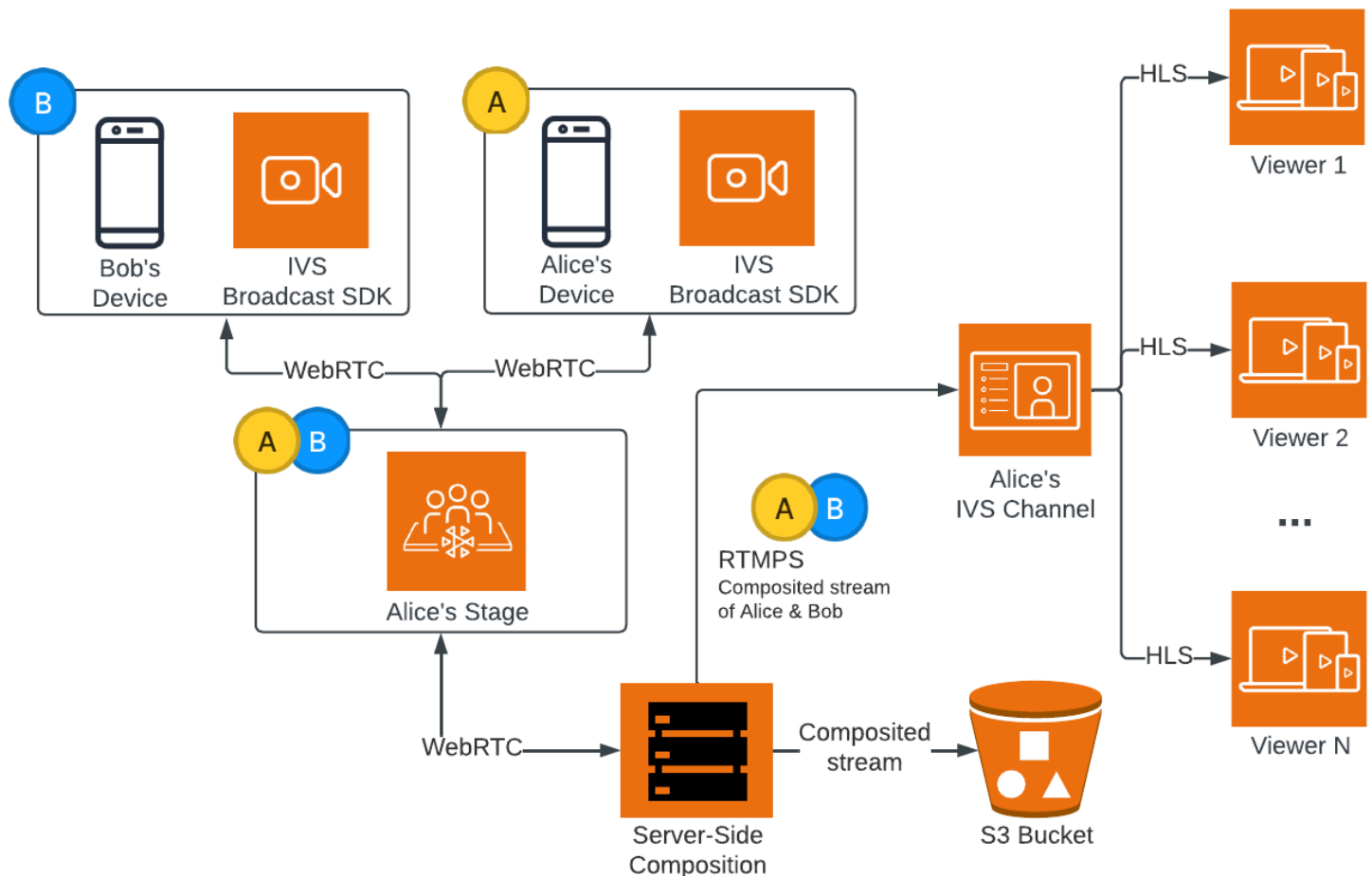
```
{
  "version": "0",
  "id": "12345678-1a23-4567-a1bc-1a2b34567890",
  "detail-type": "IVS Stage Update",
  "source": "aws.ivs",
  "account": "123456789012",
  "time": "2020-06-23T20:12:36Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:ivs:us-west-2:123456789012:stage/AbCdef1G2hij"
  ],
  "detail": {
    "session_id": "st-1234567890",
    "event_name": "Participant Unpublished",
    "user_id": "Your User Id",
    "participant_id": "xYz1c2d3e4f"
  }
}
```


Komposisi Sisi Server (Streaming Waktu Nyata)

Komposisi sisi server menggunakan server IVS untuk mencampur audio dan video dari semua peserta panggung dan kemudian mengirimkan video campuran ini ke saluran IVS (misalnya, untuk menjangkau audiens yang lebih besar) atau bucket S3. Komposisi sisi server dipanggil melalui titik akhir bidang kontrol IVS di wilayah asal panggung.

Menyiarkan atau merekam panggung menggunakan komposisi sisi server menawarkan banyak manfaat, menjadikannya pilihan yang menarik bagi pengguna yang mencari alur kerja video berbasis cloud yang efisien dan andal.

Diagram ini menggambarkan cara kerja komposisi sisi server:



ManfaatIT

Dibandingkan dengan komposisi sisi klien, komposisi sisi server memiliki manfaat sebagai berikut:

- Mengurangi beban klien - Dengan komposisi sisi server, beban pemrosesan dan penggabungan sumber audio dan video digeser dari perangkat klien individual ke server itu sendiri. Komposisi sisi server menghilangkan kebutuhan perangkat klien untuk menggunakan CPU dan sumber daya jaringan mereka untuk menyusun tampilan dan mengirimkannya ke IVS. Ini berarti pemirsa dapat menonton siaran tanpa perangkat mereka harus menangani tugas intensif sumber daya, yang dapat meningkatkan masa pakai baterai dan pengalaman menonton yang lebih lancar.
- Kualitas yang konsisten — Komposisi sisi server memungkinkan kontrol yang tepat atas kualitas, resolusi, dan bitrate aliran akhir. Ini memastikan pengalaman menonton yang konsisten untuk semua pemirsa, terlepas dari kemampuan perangkat masing-masing.
- Ketahanan — Dengan memusatkan proses komposisi di server, siaran menjadi lebih kuat. Bahkan jika perangkat penerbit mengalami keterbatasan teknis atau fluktuasi, server dapat beradaptasi dan memberikan aliran yang lebih lancar ke semua audiens.
- Efisiensi bandwidth — Karena server menangani komposisi, penerbit panggung tidak perlu menghabiskan bandwidth ekstra untuk menyiarkan video ke IVS.

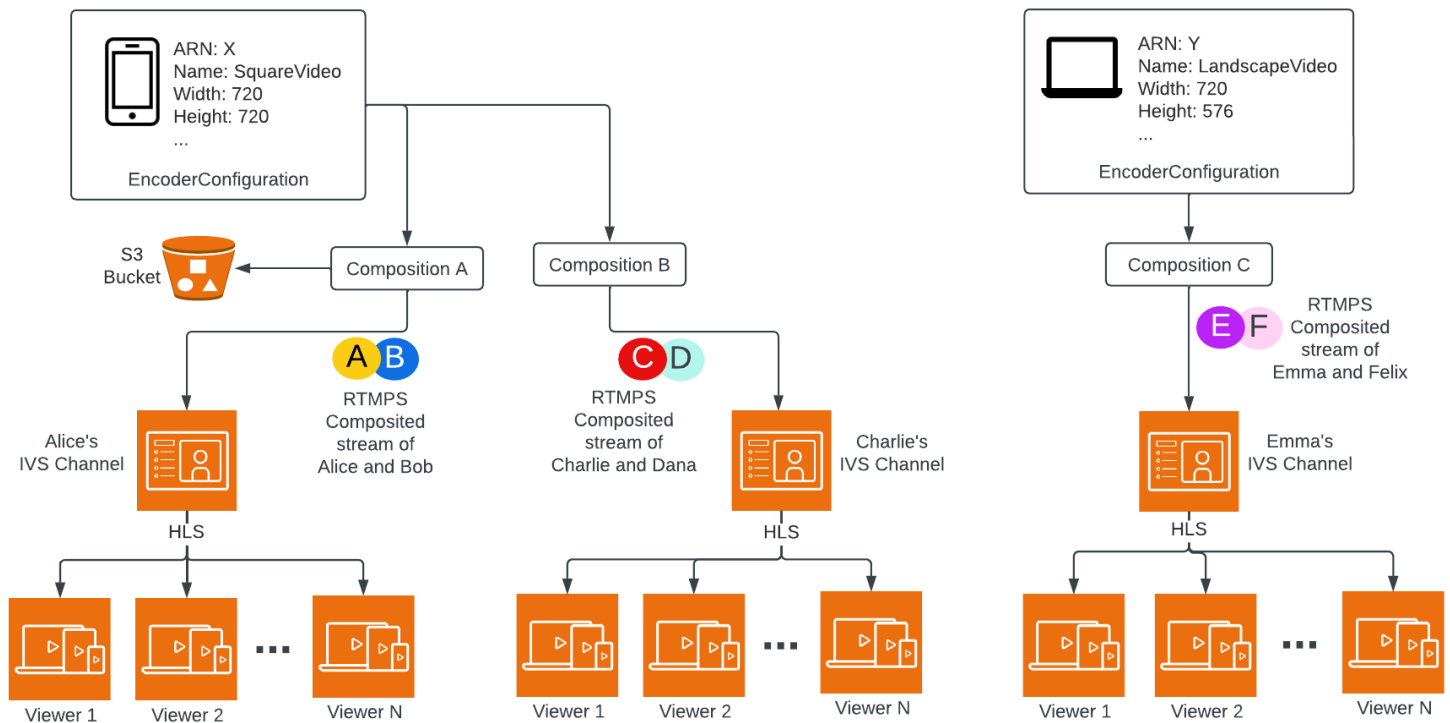
Atau, untuk menyiarkan panggung ke saluran IVS, Anda dapat melakukan komposisi sisi klien; lihat [Mengaktifkan Beberapa Host pada Aliran IVS di Panduan Pengguna Streaming Latensi Rendah IVS](#).

IVS API

Komposisi sisi server menggunakan elemen API utama ini:

- Sebuah `EncoderConfiguration` objek memungkinkan Anda untuk menyesuaikan format video yang akan dihasilkan (tinggi, lebar, bitrate, dan parameter streaming lainnya). Anda dapat menggunakan kembali `EncoderConfiguration` setiap kali Anda memanggil titik `StartComposition` akhir.
- Titik akhir komposisi melacak komposisi dan output video ke saluran IVS.
- `StorageConfiguration` melacak ember S3 tempat komposisi direkam.

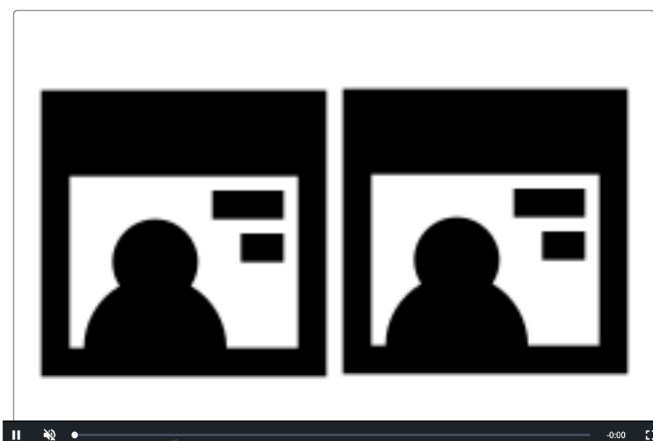
Untuk menggunakan komposisi sisi server, Anda perlu membuat `EncoderConfiguration` dan melampirkannya saat memanggil titik akhir `StartComposition`. Dalam contoh ini, `SquareVideoEncoderConfiguration` digunakan dalam dua Komposisi:



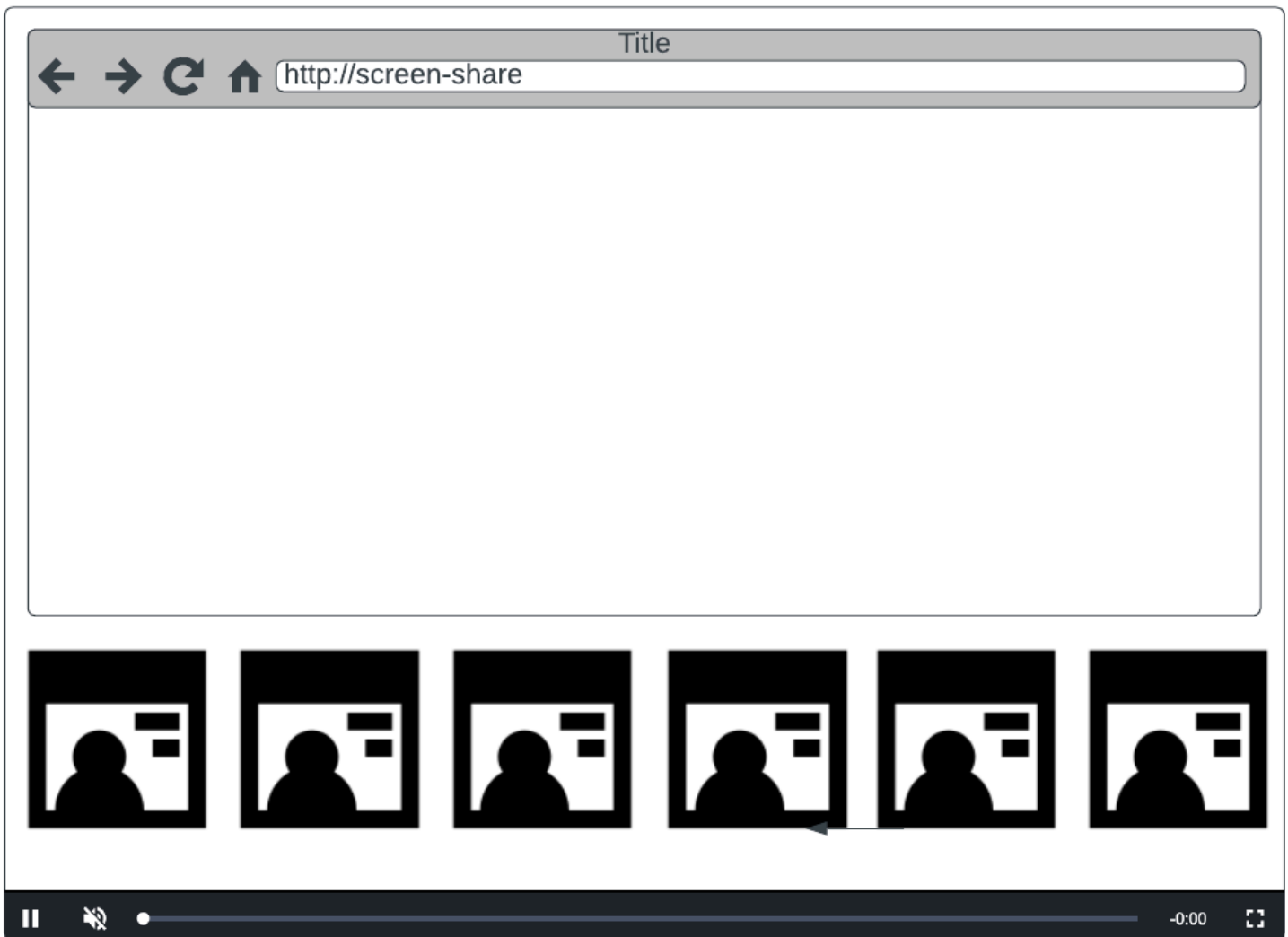
Untuk informasi selengkapnya, lihat [Referensi API Streaming Waktu Nyata IVS](#).

Layout

Secara default, fitur komposisi sisi server menggunakan tata letak kisi untuk mengatur peserta panggung dalam slot berukuran sama:



Tata letak ini menyediakan opsi bagi pelanggan untuk mengkonfigurasi dan memanggil slot unggulan. Slot unggulan ada di layar utama, dengan peserta lain ditampilkan di bawahnya dalam slot berukuran sama:



Catatan: Resolusi maksimum yang didukung oleh penerbit panggung pada komposisi sisi server adalah 1080p. Jika penerbit mengirimkan video yang lebih tinggi dari 1080p, penerbit akan ditampilkan sebagai peserta khusus audio.

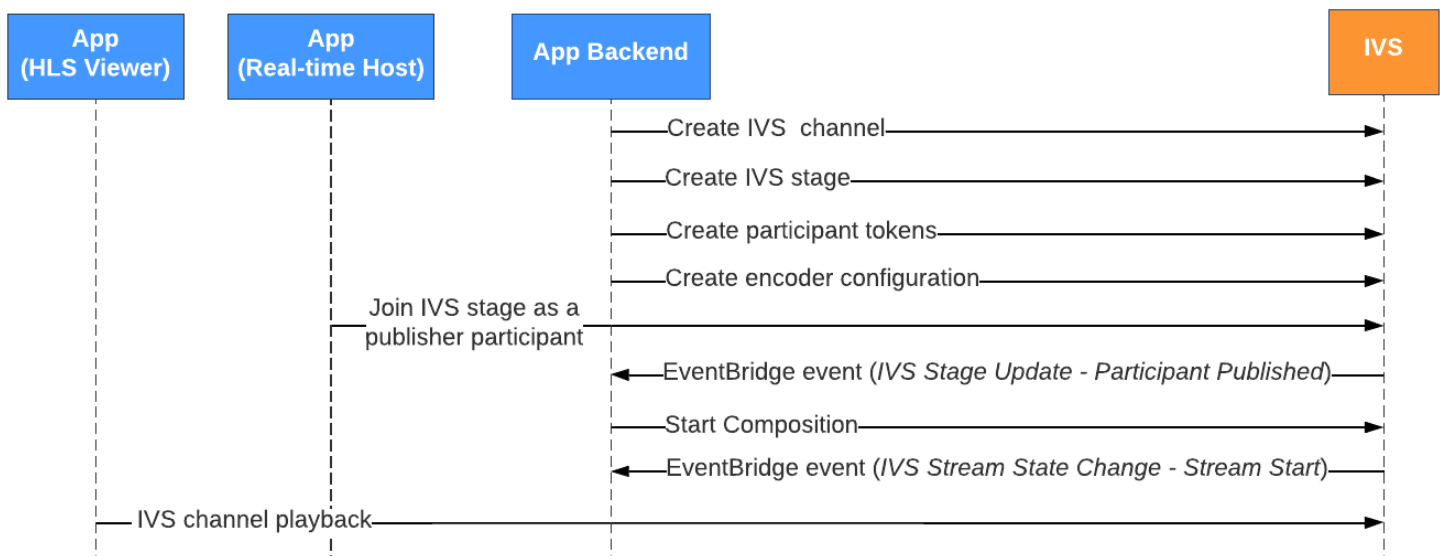
Memulai

Prasyarat

Untuk menggunakan komposisi sisi server, Anda harus memiliki panggung dengan penerbit aktif dan menggunakan saluran IVS dan/atau bucket S3 sebagai tujuan komposisi. Di bawah ini, kami menjelaskan satu kemungkinan alur kerja yang menggunakan EventBridge peristiwa untuk memulai komposisi yang menyiarkan panggung ke saluran IVS saat peserta menerbitkan. Atau, Anda dapat memulai dan menghentikan komposisi berdasarkan logika aplikasi Anda sendiri. Lihat [Perekaman](#)

[Komposit](#) untuk contoh lain yang menampilkan penggunaan komposisi sisi server untuk merekam panggung langsung ke bucket S3.

1. Buat saluran IVS. Lihat [Memulai dengan Streaming Latensi Rendah Amazon IVS](#).
2. Buat tahap IVS dan token peserta untuk setiap penerbit.
3. Buat [EncoderConfiguration](#).
4. Bergabunglah dengan panggung dan publikasikan ke sana. (Lihat bagian “Penerbitan dan Berlangganan” dari panduan SDK siaran streaming real-time: [Web](#), [Android](#), dan [iOS](#).)
5. Saat Anda menerima EventBridge acara Peserta yang Diterbitkan, hubungi [StartComposition](#).
6. Tunggu beberapa detik dan lihat tampilan gabungan di pemutaran saluran.



Catatan: Komposisi melakukan shutdown otomatis setelah 60 detik tidak aktif dari peserta penerbit di atas panggung. Pada saat itu, Komposisi dihentikan dan transisi ke keadaan. STOPPED Komposisi secara otomatis dihapus setelah beberapa menit di STOPPED negara bagian.

Instruksi CLI

Menggunakan AWS CLI adalah opsi lanjutan dan mengharuskan Anda mengunduh dan mengonfigurasi CLI terlebih dahulu di mesin Anda. Untuk detailnya, lihat [Panduan Pengguna Antarmuka Baris Perintah AWS](#).

Sekarang Anda dapat menggunakan CLI untuk membuat dan mengelola sumber daya. Titik akhir Komposisi berada di bawah `ivs-realtime` namespace.

Buat Sumber EncoderConfiguration Daya

An EncoderConfiguration adalah objek yang memungkinkan Anda untuk menyesuaikan format video yang dihasilkan (tinggi, lebar, bitrate, dan parameter streaming lainnya). Anda dapat menggunakan kembali EncoderConfiguration setiap kali Anda memanggil titik akhir Komposisi, seperti yang dijelaskan pada langkah berikutnya.

Perintah di bawah ini membuat EncoderConfiguration sumber daya yang mengonfigurasi parameter komposisi video sisi server seperti bitrate video, kecepatan bingkai, dan resolusi:

```
aws ivs-realtime create-encoder-configuration --name "MyEncoderConfig" --video
"bitrate=2500000,height=720,width=1280,framerate=30"
```

Tanggapannya adalah:

```
{
  "encoderConfiguration": {
    "arn": "arn:aws:ivs:us-east-1:927810967299:encoder-configuration/9W590BY2M8s4",
    "name": "MyEncoderConfig",
    "tags": {},
    "video": {
      "bitrate": 2500000,
      "framerate": 30,
      "height": 720,
      "width": 1280
    }
  }
}
```

Mulai Komposisi

Menggunakan EncoderConfiguration ARN yang disediakan dalam respons di atas, buat sumber daya Komposisi Anda:

```
aws ivs-realtime start-composition --stage-arn "arn:aws:ivs:us-
east-1:927810967299:stage/8faHz1SQp0ik" --destinations '[{"channel": {"channelArn":
"arn:aws:ivs:us-east-1:927810967299:channel/D01MW4dfMR8r", "encoderConfigurationArn":
"arn:aws:ivs:us-east-1:927810967299:encoder-configuration/9W590BY2M8s4"}}]'
```

Tanggapan akan menunjukkan bahwa Komposisi dibuat dengan STARTING status. Setelah Komposisi mulai menerbitkan komposisi, status bertransisi keACTIVE. (Anda dapat melihat status dengan memanggil GetComposition titik akhir ListCompositions atau.)

Setelah KomposisiACTIVE, tampilan komposit tahap IVS terlihat di saluran IVS, menggunakan: ListCompositions

```
aws ivs-realtime list-compositions
```

Tanggapannya adalah:

```
{
  "compositions": [
    {
      "arn": "arn:aws:ivs:us-east-1:927810967299:composition/YVoaXkKdEdRP",
      "destinations": [
        {
          "id": "bD9rRoN91fHU",
          "startTime": "2023-09-21T15:38:39+00:00",
          "state": "ACTIVE"
        }
      ],
      "stageArn": "arn:aws:ivs:us-east-1:927810967299:stage/8faHz1SQp0ik",
      "startTime": "2023-09-21T15:38:37+00:00",
      "state": "ACTIVE",
      "tags": {}
    }
  ]
}
```

Catatan: Anda harus memiliki peserta penerbit yang aktif menerbitkan ke panggung untuk menjaga komposisi tetap hidup. [Untuk informasi selengkapnya, lihat bagian “Menerbitkan dan Berlangganan” dari panduan SDK siaran streaming real-time: Web, Android, dan iOS.](#) Anda harus membuat token panggung yang berbeda untuk setiap peserta.

Aktifkan Berbagi Layar

Untuk menggunakan tata letak berbagi layar tetap, ikuti langkah-langkah di bawah ini.



```

{
  "composition" : {
    "arn" : "arn:aws:ivs:us-east-1:927810967299:composition/B19tQcXRgtoz",
    "destinations" : [ {
      "configuration" : {
        "channel" : {
          "channelArn" : "arn:aws:ivs:us-east-1:927810967299:channel/
D0lMW4dfMR8r",
          "encoderConfigurationArn" : "arn:aws:ivs:us-east-1:927810967299:encoder-
configuration/DEkQHWPVa0w0"
        },
        "name" : ""
      },
      "id" : "SGmgBXTULuXv",
      "state" : "STARTING"
    } ],
    "layout" : {
      "grid" : {
        "featuredParticipantAttribute" : "screen-share"
      }
    },
    "stageArn" : "arn:aws:ivs:us-east-1:927810967299:stage/8faHz1SQp0ik",
    "startTime" : "2023-09-27T21:32:38Z",
    "state" : "STARTING",
    "tags" : { }
  }
}

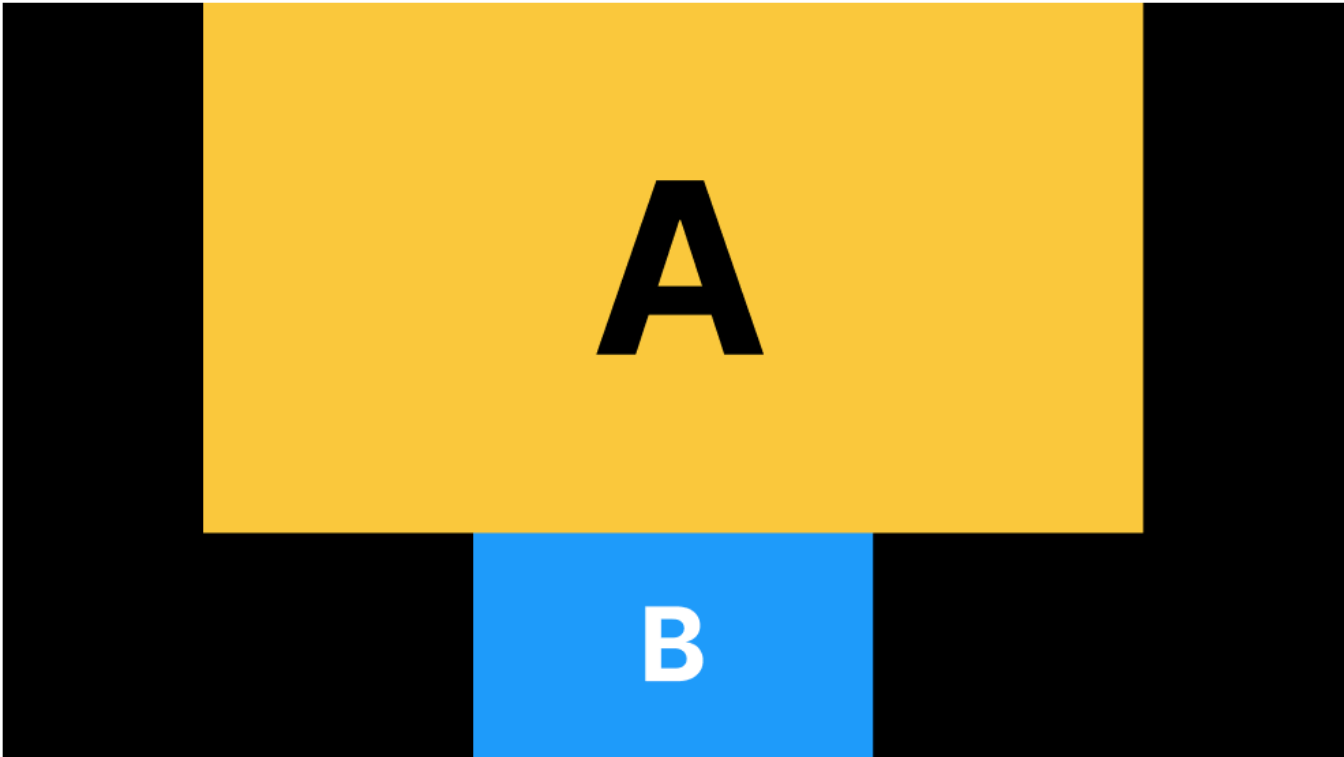
```

Ketika peserta panggung E813MFk1PWLF bergabung dengan panggung, video peserta tersebut akan ditampilkan di slot unggulan, dan semua penerbit panggung lainnya akan ditampilkan di bawah slot:

Channel details

Channel name test-channel	Channel type Standard	Video latency Low
Playback authorization Disabled	Auto-record to S3 Disabled	ARN 

▼ Live stream



Note: Playback will consume resources, and you will incur live video output cost. [Learn more](#)

State LIVE	Health ✔ Healthy	Duration 00:00:08	Viewers 0
----------------------	---------------------	----------------------	--------------

► Timed Metadata

Hentikan Komposisi

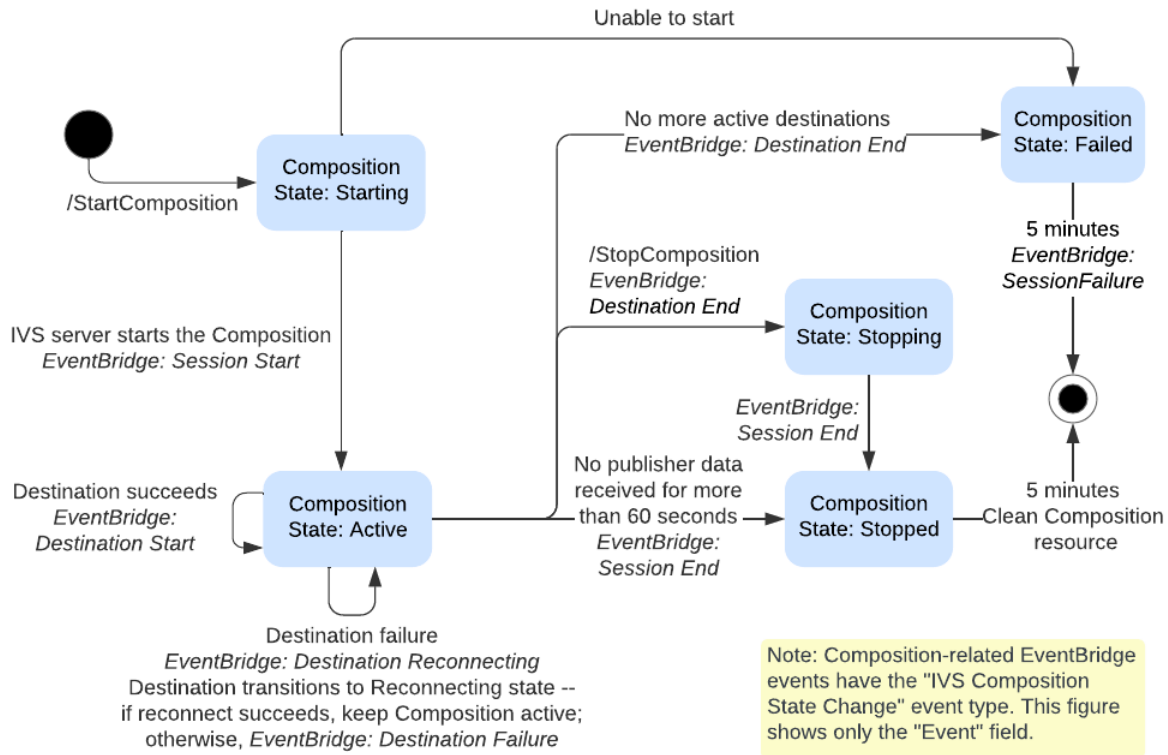
Untuk menghentikan komposisi di titik mana pun, hubungi `StopComposition` titik akhir:

```
aws ivs-realtime stop-composition --arn arn:aws:ivs:us-east-1:927810967299:composition/B19tQcXRgtoz
```

Siklus Hidup Komposisi

Gunakan diagram di bawah ini untuk memahami transisi keadaan dari Komposisi. Pada tingkat tinggi, siklus hidup suatu Komposisi adalah sebagai berikut:

1. Sumber daya Komposisi dibuat saat pengguna memanggil titik StartComposition akhir
2. Setelah IVS berhasil memulai Komposisi, EventBridge acara “Perubahan Status Komposisi IVS (Mulai Sesi)” dikirim. Lihat [Menggunakan EventBridge dengan IVS Real-Time Streaming](#) untuk detail tentang acara.
3. Setelah Komposisi dalam keadaan aktif, hal berikut dapat terjadi:
 - Pengguna menghentikan Komposisi - Jika StopComposition titik akhir dipanggil, IVS memulai penutupan Komposisi yang anggun, mengirimkan peristiwa “Akhir Tujuan” diikuti dengan acara “Akhir Sesi”.
 - Komposisi melakukan shutdown otomatis - Jika tidak ada peserta yang aktif mempublikasikan ke tahap IVS, Komposisi diselesaikan secara otomatis setelah 60 detik dan EventBridge acara dikirim.
 - Kegagalan tujuan — Jika tujuan tiba-tiba gagal (misalnya, saluran IVS akan dihapus), tujuan transisi ke RECONNECTING status dan acara “Destination Reconnecting” akan dikirim. Jika pemulihan tidak mungkin, IVS mentransisikan tujuan ke FAILED negara bagian dan acara “Kegagalan Tujuan” dikirim. IVS membuat komposisi tetap hidup jika setidaknya salah satu tujuannya aktif.
4. Setelah komposisi dalam FAILED keadaan STOPPED atau, secara otomatis dibersihkan setelah lima menit. (Maka tidak lagi diambil oleh ListCompositions atau GetComposition.)



Rekaman Komposit (Streaming Waktu Nyata)

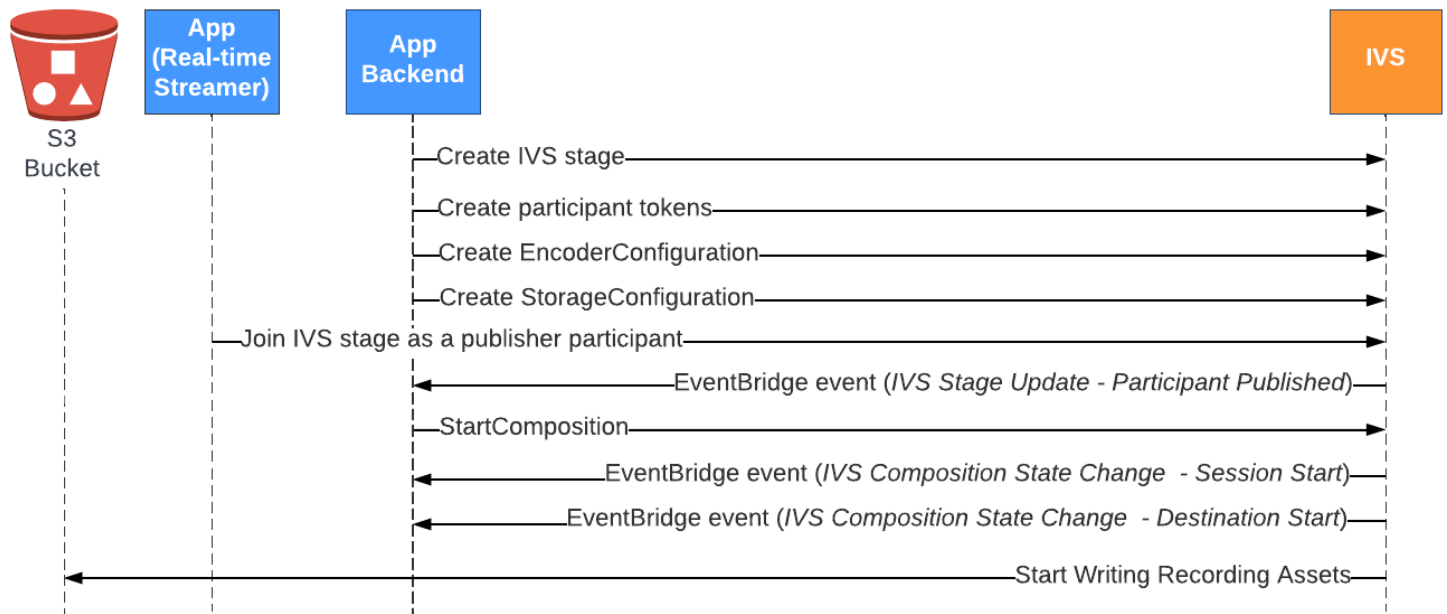
[Dokumen ini menjelaskan cara menggunakan fitur perekaman komposit dalam komposisi sisi server.](#)

Rekaman komposit memungkinkan Anda menghasilkan rekaman HLS dari tahap IVS dengan menggabungkan semua penerbit panggung secara efektif ke dalam satu tampilan menggunakan server IVS, dan kemudian menyimpan video yang dihasilkan ke ember S3.

Prasyarat

Untuk menggunakan rekaman komposit, Anda harus memiliki panggung dengan penerbit aktif dan bucket S3 untuk digunakan sebagai tujuan perekaman. Di bawah ini, kami menjelaskan satu kemungkinan alur kerja yang menggunakan EventBridge peristiwa untuk merekam komposisi ke bucket S3. Atau, Anda dapat memulai dan menghentikan komposisi berdasarkan logika aplikasi Anda sendiri.

1. Buat [tahap IVS](#) dan token peserta untuk setiap penerbit.
2. Buat [EncoderConfiguration](#) (objek yang mewakili bagaimana video yang direkam harus dirender).
3. Buat [ember S3](#) dan a [StorageConfiguration](#) (tempat konten rekaman akan disimpan).
4. [Bergabunglah dengan panggung dan publikasikan ke sana.](#)
5. Saat Anda menerima [EventBridge acara](#) Peserta Published, hubungi [StartComposition](#) dengan [DestinationConfiguration](#) objek S3 sebagai tujuan
6. Setelah beberapa detik, Anda akan dapat melihat segmen HLS disimpan ke ember S3 Anda.



Catatan: Komposisi melakukan shutdown otomatis setelah 60 detik tidak aktif dari peserta penerbit di atas panggung. Pada saat itu, komposisi dihentikan dan transisi ke keadaan. STOPPED Komposisi secara otomatis dihapus setelah beberapa menit di STOPPED negara bagian. Untuk detailnya, lihat [Siklus Hidup Komposisi](#) dalam Komposisi Sisi Server.

Contoh Perekaman Komposit: StartComposition dengan Tujuan Bucket S3

Contoh di bawah ini menunjukkan panggilan tipikal ke [StartComposition](#) titik akhir, menentukan S3 sebagai satu-satunya tujuan untuk komposisi. Setelah komposisi bertransisi ke ACTIVE status, segmen video dan metadata akan mulai ditulis ke bucket S3 yang ditentukan oleh objek. `storageConfiguration` Untuk membuat komposisi dengan layout yang berbeda, lihat “Layouts” di [Server-Side Composition](#) dan [IVS Real-Time Streaming API Reference](#).

Permintaan

```

POST /StartComposition HTTP/1.1
Content-type: application/json

{
  "destinations": [
    {
      "s3": {
        "encoderConfigurationArns": [
          "arn:aws:ivs:ap-northeast-1:927810967299:encoder-configuration/
PAAwglkRtjge"
        ]
      }
    }
  ]
}
  
```

```

    ],
    "storageConfigurationArn": "arn:aws:ivs:ap-
northeast-1:927810967299:storage-configuration/ZBcEbgE24Cq"
  }
}
],
"idempotencyToken": "db1i782f1g9",
"stageArn": "arn:aws:ivs:ap-northeast-1:927810967299:stage/WyGkzNFGwiwr"
}

```

Respons

```

{
  "composition": {
    "arn": "arn:aws:ivs:ap-northeast-1:927810967299:composition/s2AdaGubvQgp",
    "destinations": [
      {
        "configuration": {
          "name": "",
          "s3": {
            "encoderConfigurationArns": [
              "arn:aws:ivs:ap-northeast-1:927810967299:encoder-
configuration/PAAwglkRtjge"
            ],
            "recordingConfiguration": {
              "format": "HLS"
            },
            "storageConfigurationArn": "arn:aws:ivs:ap-
northeast-1:927810967299:storage-configuration/ZBcEbgE24Cq"
          }
        },
        "detail": {
          "s3": {
            "recordingPrefix": "MNALAcH9j2EJ/s2AdaGubvQgp/2pBRKrNgX1ff/
composite"
          }
        },
        "id": "2pBRKrNgX1ff",
        "state": "STARTING"
      }
    ],
    "layout": null,
    "stageArn": "arn:aws:ivs:ap-northeast-1:927810967299:stage/WyGkzNFGwiwr",
  }
}

```

```
    "startTime": "2023-11-01T06:25:37Z",
    "state": "STARTING",
    "tags": {}
  }
}
```

`recordingPrefixBidang`, yang ada dalam `StartComposition` respons dapat digunakan untuk menentukan di mana konten rekaman akan disimpan.

Isi Rekaman

Saat komposisi bertransisi ke `ACTIVE` status, Anda akan mulai melihat segmen video HLS dan file metadata yang ditulis ke bucket S3 yang disediakan saat menelepon. `StartComposition` Konten ini tersedia untuk pasca-pemrosesan atau pemutaran sebagai video sesuai permintaan.

Perhatikan bahwa setelah komposisi menjadi live, peristiwa “Perubahan Status Komposisi IVS” dipancarkan, dan mungkin perlu sedikit waktu sebelum file manifes dan segmen video ditulis. Kami menyarankan Anda memutar ulang atau memproses streaming yang direkam hanya setelah acara “Perubahan Status Komposisi IVS (Akhir Sesi)” diterima. Untuk detailnya, lihat [Menggunakan EventBridge dengan IVS Real-Time Streaming](#).

Berikut ini adalah struktur direktori sampel dan isi rekaman sesi IVS langsung:

```
MNALAch9j2EJ/s2AdaGubvQgp/2pBRK1rNgX1ff/composite
  events
    recording-started.json
    recording-ended.json
  media
    hls
```

`eventsFolder` berisi file metadata yang sesuai dengan peristiwa perekaman. File metadata JSON dihasilkan saat perekaman dimulai, berakhir dengan sukses, atau diakhiri dengan kegagalan:

- `events/recording-started.json`
- `events/recording-ended.json`
- `events/recording-failed.json`

`eventsFolder` tertentu akan berisi `recording-started.json` dan salah satu `recording-ended.json` atau `recording-failed.json`.

Ini berisi metadata yang terkait dengan sesi rekaman dan format outputnya. Rincian JSON diberikan di bawah ini.

`mediaFolder` berisi konten media yang didukung. `hlsSubfolder` berisi semua media dan file manifes yang dihasilkan selama sesi komposisi dan dapat dimainkan dengan pemutar IVS. Manifes HLS terletak di `multivariant.m3u8` folder.

Kebijakan Bucket untuk StorageConfiguration

Saat `StorageConfiguration` objek dibuat, IVS akan mendapatkan akses untuk menulis konten ke bucket S3 yang ditentukan. Akses ini diberikan dengan melakukan modifikasi pada kebijakan bucket S3. Jika kebijakan untuk bucket diubah dengan cara yang menghapus akses IVS, rekaman yang sedang berlangsung dan baru akan gagal.

Contoh di bawah ini menunjukkan kebijakan bucket S3 yang memungkinkan IVS menulis ke bucket S3:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CompositeWrite-y1d212y",
      "Effect": "Allow",
      "Principal": {
        "Service": "ivs-composite.ap-northeast-1.amazonaws.com"
      },
      "Action": [
        "s3:PutObject",
        "s3:PutObjectAcl"
      ],
      "Resource": "arn:aws:s3:::my-s3-bucket/*",
      "Condition": {
        "StringEquals": {
          "s3:x-amz-acl": "bucket-owner-full-control"
        },
        "Bool": {
          "aws:SecureTransport": "true"
        }
      }
    }
  ]
}
```

}

File Metadata JSON

Metadata ini dalam format JSON. Ini terdiri dari informasi berikut:

Bidang	Tipe	Diperlukan	Deskripsi
stage_arn	string	Ya	ARN dari panggung yang digunakan sebagai sumber komposisi.
media	objek	Ya	Objek yang berisi objek yang disebutkan dari konten media yang tersedia untuk rekaman ini. Nilai yang valid: "hls".
hls	objek	Ya	Bidang yang disebutkan yang menjelaskan output format Apple HLS.
duration_ms	integer	Bersyarat	Durasi konten HLS yang direkam dalam milidetik. Ini hanya tersedia jika <code>recording_status</code> ada "RECORDING_ENDED" atau "RECORDING_ENDED_WITH_FAILURE" . Jika terjadi kegagalan sebelum perekaman dilakukan, ini adalah 0.
path	string	Ya	Jalur relatif dari awalan S3 tempat konten HLS disimpan.
playlist	string	Ya	Nama file daftar putar master HLS.

Bidang	Tipe	Diperlukan	Deskripsi
<code>renditions</code>	objek	Ya	Array rendisi (varian HLS) objek metadata. Selalu ada setidaknya satu rendisi.
<code>path</code>	string	Ya	Jalur relatif dari awalan S3 tempat konten HLS disimpan untuk rendisi ini.
<code>playlist</code>	string	Ya	Nama file playlist media untuk rendisi ini.
<code>resolution_height</code>	int	Bersyarat	Tinggi resolusi piksel dari video yang dikodekan. Ini hanya tersedia ketika rendisi berisi trek video.
<code>resolution_width</code>	int	Bersyarat	Lebar resolusi piksel dari video yang dikodekan. Ini hanya tersedia ketika rendisi berisi trek video.
<code>recording_ended_at</code>	string	Bersyarat	Stempel waktu RFC 3339 UTC saat rekaman berakhir. Ini hanya tersedia jika <code>recording_status</code> ada "RECORDING_ENDED" atau "RECORDING_ENDED_WITH_FAILURE". <code>recording_started_at</code> dan <code>recording_ended_at</code> merupakan stempel waktu saat peristiwa ini dihasilkan dan mungkin tidak sama persis dengan stempel waktu segmen video HLS. Untuk menentukan durasi rekaman secara akurat, gunakan <code>duration_ms</code> bidang.

Bidang	Tipe	Diperlukan	Deskripsi
<code>recording_started_at</code>	string	Bersyarat	Stempel waktu RFC 3339 UTC saat perekaman dimulai. Ini tidak tersedia kapan <code>recording_status</code> <code>RECORDING_STARTED_FAILED</code> . Lihat catatan di atas untuk <code>recording_ended_at</code> .
<code>recording_status</code>	string	Ya	Status rekaman. Nilai yang valid: <code>"RECORDING_STARTED"</code> , <code>"RECORDING_ENDED"</code> , <code>"RECORDING_START_FAILED"</code> , <code>"RECORDING_ENDED_WITH_FAILURE"</code> .
<code>recording_status_message</code>	string	Bersyarat	Informasi deskriptif tentang status. Ini hanya tersedia jika <code>recording_status</code> ada <code>"RECORDING_ENDED"</code> atau <code>"RECORDING_ENDED_WITH_FAILURE"</code> .
<code>version</code>	string	Ya	Versi skema metadata.

Contoh: recording-started.json

```
{
  "version": "v1",
  "stage_arn": "arn:aws:ivs:ap-northeast-1:123456789012:stage/aAbBcCdDeE12",
  "recording_started_at": "2023-11-01T06:01:36Z",
  "recording_status": "RECORDING_STARTED",
  "media": {
    "hls": {
      "path": "media/hls",
```

```
"playlist": "multivariant.m3u8",
"renditions": [
  {
    "path": "720p30-abcdeABCDE12",
    "playlist": "playlist.m3u8",
    "resolution_width": 1280,
    "resolution_height": 720
  }
]
```

Contoh: recording-ended.json

```
{
  "version": "v1",
  "stage_arn": "arn:aws:ivs:ap-northeast-1:123456789012:stage/aAbBcCdDeE12",
  "recording_started_at": "2023-10-27T17:00:44Z",
  "recording_ended_at": "2023-10-27T17:08:24Z",
  "recording_status": "RECORDING_ENDED",
  "media": {
    "hls": {
      "duration_ms": 460315,
      "path": "media/hls",
      "playlist": "multivariant.m3u8",
      "renditions": [
        {
          "path": "720p30-abcdeABCDE12",
          "playlist": "playlist.m3u8",
          "resolution_width": 1280,
          "resolution_height": 720
        }
      ]
    }
  }
}
```

Contoh: perekaman-failed.json

```
{
  "version": "v1",
```

```
"stage_arn": "arn:aws:ivs:ap-northeast-1:123456789012:stage/aAbBcCdDeE12",
"recording_started_at": "2023-10-27T17:00:44Z",
"recording_ended_at": "2023-10-27T17:08:24Z",
"recording_status": "RECORDING_ENDED_WITH_FAILURE",
"media": {
  "hls": {
    "duration_ms": 460315,
    "path": "media/hls",
    "playlist": "multivariant.m3u8",
    "renditions": [
      {
        "path": "720p30-abcdeABCDE12",
        "playlist": "playlist.m3u8",
        "resolution_width": 1280,
        "resolution_height": 720
      }
    ]
  }
}
```

Pemutaran Konten Rekaman dari Bucket Pribadi

Secara default, konten yang direkam bersifat pribadi; karenanya, objek ini tidak dapat diakses untuk pemutaran menggunakan URL S3 langsung. Jika Anda mencoba membuka daftar putar multivariat HLS (file m3u8) untuk pemutaran menggunakan pemutar IVS atau pemutar lain, Anda akan mendapatkan kesalahan (misalnya, “Anda tidak memiliki izin untuk mengakses sumber daya yang diminta”). Sebagai gantinya, Anda dapat memutar kembali file-file ini dengan Amazon CloudFront CDN (Jaringan Pengiriman Konten).

CloudFront distribusi dapat dikonfigurasi untuk menyajikan konten dari bucket pribadi. Biasanya ini lebih disukai daripada memiliki bucket yang dapat diakses secara terbuka di mana pembacaan melewati kontrol yang ditawarkan oleh CloudFront Anda dapat mengatur distribusi agar dilayani dari bucket pribadi dengan membuat kontrol akses asal (OAC), yang merupakan CloudFront pengguna khusus yang memiliki izin membaca di bucket asal pribadi. Anda dapat membuat OAC setelah membuat distribusi, melalui CloudFront konsol atau API. Lihat [Membuat kontrol akses asal baru](#) di Panduan CloudFront Pengembang Amazon.

Menyiapkan Pemutaran menggunakan CloudFront CORS Diaktifkan

Contoh ini mencakup bagaimana pengembang dapat mengatur CloudFront distribusi dengan CORS diaktifkan, memungkinkan pemutaran rekaman mereka dari domain apa pun. Ini sangat berguna selama fase pengembangan, tetapi Anda dapat memodifikasi contoh di bawah ini agar sesuai dengan kebutuhan produksi Anda.

Langkah 1: Buat Bucket S3

Buat bucket S3 yang akan digunakan untuk menyimpan rekaman. Perhatikan bahwa bucket harus berada di wilayah yang sama dengan yang Anda gunakan untuk alur kerja IVS Anda.

Tambahkan kebijakan CORS permisif ke bucket:

1. Di konsol AWS, buka tab S3 Bucket Permissions.
2. Salin kebijakan CORS di bawah ini dan tempel di bawah Cross-origin resource sharing (CORS). Ini akan memungkinkan akses CORS pada bucket S3.

```
[
  {
    "AllowedHeaders": [
      "*"
    ],
    "AllowedMethods": [
      "PUT",
      "POST",
      "DELETE",
      "GET"
    ],
    "AllowedOrigins": [
      "*"
    ],
    "ExposeHeaders": [
      "x-amz-server-side-encryption",
      "x-amz-request-id",
      "x-amz-id-2"
    ]
  }
]
```

Langkah 2: Buat CloudFront Distribusi

Lihat [Membuat CloudFront distribusi](#) di Panduan CloudFront Pengembang.

Menggunakan konsol AWS, masukkan informasi berikut:

Untuk bidang ini...	Pilih ini...
Domain Asal	Bucket S3 dibuat pada langkah sebelumnya
Akses Asal	Pengaturan kontrol akses asal (disarankan), menggunakan parameter default
Perilaku cache default: Kebijakan Protokol Penampil	Arahkan ulang HTTP ke HTTPS
Perilaku cache default: Metode HTTP yang diizinkan	DAPATKAN, KEPALA, dan OPSI
Perilaku cache default: Kunci cache dan permintaan asal	CachingDisabled kebijakan
Perilaku cache default: Kebijakan permintaan asal	CORS-S3asal
Perilaku cache default: Kebijakan header respons	SimpleCORS
Firewall Aplikasi Web	Aktifkan perlindungan keamanan

Kemudian simpan CloudFront distribusinya.

Langkah 3: Siapkan Kebijakan Bucket S3

1. Hapus semua StorageConfiguration yang telah Anda atur untuk bucket S3. Ini akan menghapus kebijakan bucket apa pun yang ditambahkan secara otomatis saat membuat kebijakan untuk bucket tersebut.

2. Buka CloudFront Distribusi Anda, pastikan semua bidang distribusi berada di status yang ditentukan pada langkah sebelumnya, dan Salin Kebijakan Bucket (gunakan tombol Salin kebijakan).
3. Pergi ke ember S3 Anda. Pada tab Izin, pilih Edit Kebijakan Bucket dan tempel kebijakan bucket yang Anda salin di langkah sebelumnya. Setelah langkah ini, kebijakan bucket harus memiliki CloudFront kebijakan secara eksklusif.
4. Buat StorageConfiguration, tentukan bucket S3.

Setelah StorageConfiguration dibuat, Anda akan melihat dua item dalam kebijakan bucket S3, satu memungkinkan CloudFront untuk membaca konten dan satu lagi memungkinkan IVS untuk menulis konten. Contoh kebijakan bucket final, with CloudFront dan akses IVS, ditampilkan di [Contoh: Kebijakan Bucket S3 dengan CloudFront dan Akses IVS](#).

Langkah 4: Putar Kembali Rekaman

Setelah berhasil mengatur CloudFront distribusi dan memperbarui kebijakan bucket, Anda harus dapat memutar ulang rekaman menggunakan pemutar IVS:

1. Berhasil memulai Komposisi dan pastikan Anda memiliki rekaman yang tersimpan di bucket S3.
2. Setelah mengikuti Langkah 1 hingga Langkah 3 dalam contoh ini, file video harus tersedia untuk dikonsumsi melalui CloudFront URL. CloudFront URL Anda adalah nama domain Distribusi pada tab Detail di CloudFront konsol Amazon. Seharusnya seperti ini:

```
a1b23cdef4ghij.cloudfront.net
```

3. Untuk memutar video yang direkam melalui CloudFront distribusi, temukan kunci objek untuk `multivariant.m3u8` file Anda di bawah bucket s3. Seharusnya seperti ini:

```
FDew6Szq5iTt/9NIpWJHj0wPT/fjFKbylPb3k4/composite/media/hls/  
multivariant.m3u8
```

4. Tambahkan kunci objek ke akhir CloudFront URL Anda. URL akhir Anda akan menjadi seperti ini:

```
https://a1b23cdef4ghij.cloudfront.net/FDew6Szq5iTt/9NIpWJHj0wPT/  
fjFKbylPb3k4/composite/media/hls/multivariant.m3u8
```

5. Anda sekarang dapat menambahkan URL akhir ke atribut sumber pemutar IVS untuk menonton rekaman lengkap. Untuk menonton video yang direkam, Anda dapat menggunakan demo di [Memulai](#) di IVS Player SDK: Panduan Web.

Contoh: Kebijakan Bucket S3 dengan CloudFront dan Akses IVS

Cuplikan di bawah ini mengilustrasikan kebijakan bucket S3 yang memungkinkan membaca konten CloudFront ke bucket pribadi dan IVS untuk menulis konten ke bucket. Catatan: Jangan menyalin dan menempelkan cuplikan di bawah ini ke ember Anda sendiri. Kebijakan Anda harus berisi ID yang relevan dengan CloudFront distribusi Anda dan StorageConfiguration.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CompositeWrite-7eiKaIGkC9D0",
      "Effect": "Allow",
      "Principal": {
        "Service": "ivs-composite.ap-northeast-1.amazonaws.com"
      },
      "Action": [
        "s3:PutObject",
        "s3:PutObjectAcl"
      ],
      "Resource": "arn:aws:s3:::eicheane-test-1026-2-ivs-recordings/*",
      "Condition": {
        "StringEquals": {
          "s3:x-amz-acl": "bucket-owner-full-control"
        },
        "Bool": {
          "aws:SecureTransport": "true"
        }
      }
    },
    {
      "Sid": "AllowCloudFrontServicePrincipal",
      "Effect": "Allow",
      "Principal": {
        "Service": "cloudfront.amazonaws.com"
      },
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::eicheane-test-1026-2-ivs-recordings/*",
      "Condition": {
        "StringEquals": {
          "AWS:SourceArn": "arn:aws:cloudfront::844311324168:distribution/E1NG4YMW5MN25A"
        }
      }
    }
  ]
}
```

```
    }  
  }  
]  
}
```

Memecahkan masalah

- Komposisi tidak ditulis ke bucket S3 — Pastikan bucket dan StorageConfiguration objek S3 dibuat dan berada di wilayah yang sama. Pastikan juga bahwa IVS memiliki akses ke bucket dengan memeriksa kebijakan bucket Anda; lihat [Kebijakan Bucket untuk StorageConfiguration](#).
- Saya tidak dapat menemukan komposisi saat tampil ListCompositions— Komposisi adalah sumber daya fana. Setelah mereka beralih ke keadaan akhir, mereka dihapus secara otomatis setelah beberapa menit.
- Komposisi saya berhenti secara otomatis — Komposisi akan berhenti secara otomatis jika tidak ada penerbit di panggung selama lebih dari 60 detik.

Masalah yang Diketahui

Daftar putar media yang ditulis oleh rekaman komposit memiliki tag #EXT-X-PLAYLIST-TYPE:EVENT saat komposisi sedang berlangsung. Ketika komposisi selesai, tag diperbarui ke#EXT-X-PLAYLIST-TYPE:VOD. Untuk pengalaman pemutaran yang lancar, kami sarankan Anda menggunakan daftar putar ini hanya setelah komposisi berhasil diselesaikan.

Dukungan OBS dan WHIP (Streaming Waktu Nyata)

Dokumen ini menjelaskan cara menggunakan encoder yang kompatibel dengan Whip seperti OBS untuk mempublikasikan ke streaming real-time IVS. [WHIP](#) (WebRTC-HTTP Ingestion Protocol) adalah draf IETF yang dikembangkan untuk membakukan konsumsi WebRTC.

WHIP memungkinkan kompatibilitas dengan perangkat lunak seperti OBS, menawarkan alternatif (ke SDK siaran IVS) untuk penerbitan desktop. Streamer yang lebih canggih yang akrab dengan OBS mungkin lebih menyukainya karena fitur produksinya yang canggih, seperti transisi adegan, pencampuran audio, dan grafik overlay. Ini memberi pengembang opsi serbaguna: gunakan SDK siaran web IVS untuk penerbitan browser langsung atau izinkan streamer menggunakan OBS di desktop mereka untuk alat yang lebih canggih.

Selain itu, WHIP bermanfaat dalam situasi di mana menggunakan SDK siaran IVS tidak layak atau disukai. Misalnya, dalam pengaturan yang melibatkan encoder perangkat keras, SDK siaran IVS mungkin bukan pilihan. Namun, jika encoder mendukung WHIP, Anda masih dapat mempublikasikan langsung dari encoder ke IVS.

Panduan OBS

OBS mendukung WHIP pada versi 30. [Untuk memulai, unduh OBS v30 atau yang lebih baru: https://obsproject.com/.](https://obsproject.com/)

Untuk mempublikasikan ke tahap IVS menggunakan OBS melalui WHIP, ikuti langkah-langkah berikut:

1. [Hasilkan](#) token peserta dengan kemampuan mempublikasikan. Dalam istilah WHIP, token peserta adalah token pembawa. Secara default, token peserta kedaluwarsa dalam 12 jam, tetapi Anda dapat memperpanjang durasi hingga 14 hari.
2. Klik Pengaturan. Di bagian Stream pada panel Pengaturan, pilih WHIP dari dropdown Layanan.
3. Untuk Server, masukkan <https://global.whip.live-video.net/>.
4. Untuk Token Pembawa, masukkan token peserta yang Anda buat di langkah 2.
5. Konfigurasi pengaturan video Anda seperti biasa, dengan beberapa batasan:
 - a. Streaming real-time IVS mendukung input hingga 720p pada 8,5 Mbps. Jika Anda melebihi salah satu dari batas ini, aliran Anda akan terputus.

- b. Sebaiknya atur Interval Keyframe Anda di panel Output ke 1s atau 2s. Interval keyframe yang rendah memungkinkan pemutaran video dimulai lebih cepat bagi pemirsa. Kami juga merekomendasikan pengaturan Preset Penggunaan CPU ke ultrafast dan Tune ke zerolatency, untuk mengaktifkan latensi terendah.
 - c. Karena OBS tidak mendukung simulcast, kami sarankan untuk menjaga bitrate Anda di bawah 2,5 Mbps. Hal ini memungkinkan pemirsa pada koneksi bandwidth rendah untuk menonton.
6. Tekan Mulai Streaming.

Service Quotas (Streaming Waktu Nyata)

Berikut ini adalah kuota dan batasan layanan untuk titik akhir real-time Amazon Interactive Video Service (IVS), sumber daya, dan operasi lainnya. Kuota layanan (juga dikenal sebagai batas) adalah jumlah maksimum sumber daya layanan atau operasi untuk AWS akun Anda. Artinya, batas ini per AWS akun, kecuali disebutkan lain dalam tabel. Lihat juga [AWS Service Quotas](#).

Anda menggunakan titik akhir untuk terhubung secara terprogram ke layanan. AWS Lihat juga [Titik Akhir AWS Layanan](#).

Semua kuota diberlakukan per wilayah.

Kuota Layanan Meningkatkan

Untuk kuota yang dapat disesuaikan, Anda dapat meminta kenaikan tarif melalui [AWS konsol](#). Gunakan konsol untuk melihat informasi tentang kuota layanan juga.

Kuota tingkat panggilan API tidak dapat disesuaikan.

Kuota Tingkat Panggilan API

Jenis Titik Akhir	Titik akhir	Default
Komposisi	<code>GetComposition</code>	5 TPS
Komposisi	<code>ListCompositions</code>	5 TPS
Komposisi	<code>StartComposition</code>	5 TPS
Komposisi	<code>StopComposition</code>	5 TPS
MediaEncoder	<code>CreateEncoderConfiguration</code>	5 TPS
MediaEncoder	<code>DeleteEncoderConfiguration</code>	5 TPS
MediaEncoder	<code>GetEncoderConfiguration</code>	5 TPS
MediaEncoder	<code>ListEncoderConfigurations</code>	5 TPS

Jenis Titik Akhir	Titik akhir	Default
Tahap	CreateParticipantToken	50 TPS
Tahap	CreateStage	5 TPS
Tahap	DeleteStage	5 TPS
Tahap	DisconnectParticipant	5 TPS
Tahap	GetParticipant	5 TPS
Tahap	GetStage	5 TPS
Tahap	GetStageSession	5 TPS
Tahap	ListStages	5 TPS
Tahap	UpdateStage	5 TPS
Tahap	ListParticipants	5 TPS
Tahap	ListParticipantEvents	5 TPS
Tahap	ListStageSessions	5 TPS
StorageConfiguration	CreateStorageConfiguration	5 TPS
StorageConfiguration	DeleteStorageConfiguration	5 TPS
StorageConfiguration	GetStorageConfiguration	5 TPS
StorageConfiguration	ListStorageConfigurations	5 TPS
Tanda	ListTagsForResource	10 TPS
Tanda	TagResource	10 TPS
Tanda	UntagResource	10 TPS

Kuota Lainnya

Sumber Daya atau Fitur	Default	Dapat disesuaikan	Deskripsi
EncoderConfigurations	20	Ya	Jumlah maksimum sumber daya konfigurasi encoder per akun.
Tujuan komposisi	2	Tidak	Jumlah maksimum objek Tujuan dalam sumber daya komposisi.
Komposisi: durasi maks	24	Tidak	Jumlah maksimum waktu komposisi dapat ada, dalam jam.
Komposisi	5	Ya	Sumber daya komposisi bersamaan maksimum per akun.
Peserta mempublikasikan atau berlangganan durasi	24	Tidak	Durasi maksimum waktu peserta dapat mempublikasikan atau tetap berlangganan panggung, dalam hitungan jam.
Peserta mempublikasikan resolusi	720p	Tidak	Resolusi maksimum video yang diterbitkan oleh peserta.
Bitrate unduhan peserta	8,5 Mbps	Tidak	Bitrate unduhan agregat maksimum di semua langganan peserta.
Peserta panggung (penerbit)	12	Tidak	Jumlah maksimum peserta yang dapat mempublikasikan ke panggung sekaligus.

Sumber Daya atau Fitur	Default	Dapat disesuaikan	Deskripsi
Peserta panggung (pelanggan)	10.000	Ya	Jumlah maksimum peserta yang dapat berlangganan panggung sekaligus.
Tahapan	100	Ya	Jumlah maksimum tahapan, per AWS Wilayah.

Optimasi Streaming Waktu Nyata

Untuk memastikan bahwa pengguna Anda memiliki pengalaman terbaik saat streaming dan melihat video menggunakan streaming real-time IVS, ada beberapa cara Anda dapat meningkatkan atau mengoptimalkan bagian dari pengalaman, menggunakan fitur yang kami tawarkan hari ini.

Pengantar

Saat mengoptimalkan kualitas pengalaman pengguna, penting untuk mempertimbangkan pengalaman yang diinginkan, yang dapat berubah tergantung pada konten yang mereka tonton dan kondisi jaringan.

Sepanjang panduan ini, kami fokus pada pengguna yang merupakan penerbit aliran atau pelanggan aliran, dan kami mempertimbangkan tindakan dan pengalaman yang diinginkan dari pengguna tersebut.

Streaming Adaptif: Pengkodean Berlapis dengan Simulcast

Fitur ini hanya didukung dalam versi klien berikut:

- [iOS dan Android 1.12.0 +](#)
- [Web 1.5.1 +](#)

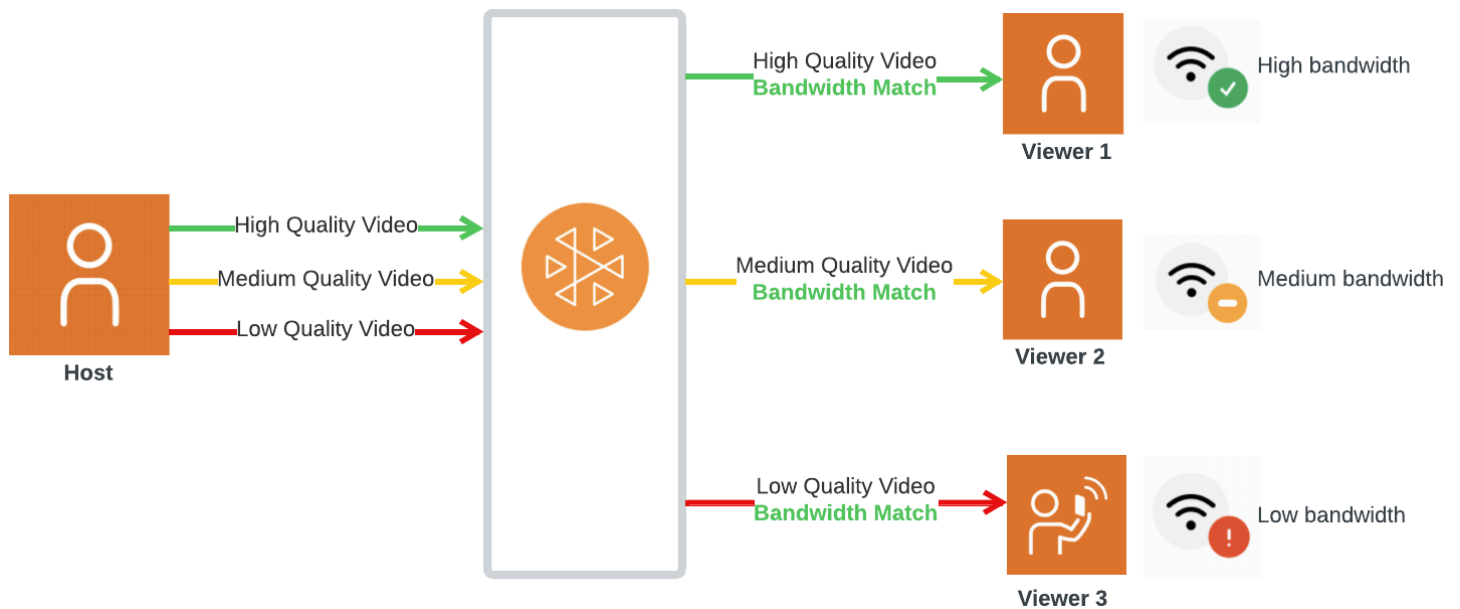
Anda harus mengirim email ke amazon-ivs-simulcast@amazon.com untuk ikut serta dalam fitur ini untuk akun Anda. Mengaktifkan simulcast melalui konfigurasi SDK tidak akan berpengaruh kecuali Anda ikut serta.

Setelah Anda memilih fitur ini, saat menggunakan [SDK siaran real-time](#) IVS, penerbit menyandikan beberapa lapisan video dan pelanggan secara otomatis beradaptasi atau mengubah kualitas yang paling sesuai untuk jaringan mereka. Kami menyebutnya encoding berlapis dengan simulcast.

Pengkodean berlapis dengan simulcast didukung di Android dan iOS, dan di browser desktop Chrome (untuk Windows dan macOS). Kami tidak mendukung encoding berlapis pada browser lain.

Pada diagram di bawah ini, host mengirimkan tiga kualitas video (tinggi, sedang, dan rendah). IVS meneruskan video berkualitas tinggi ke setiap pemirsa berdasarkan bandwidth yang tersedia; ini memberikan pengalaman optimal untuk setiap pemirsa. Jika koneksi jaringan 1 Viewer berubah dari

baik menjadi buruk, IVS secara otomatis mulai mengirim Viewer 1 video berkualitas lebih rendah, sehingga Viewer 1 dapat terus menonton streaming tanpa gangguan (dengan kualitas terbaik).



Lapisan Default, Kualitas, dan Framerate

Kualitas dan lapisan default yang disediakan untuk pengguna seluler dan web adalah sebagai berikut:

Seluler (Android, iOS)	Web (Chrome)
<p>Lapisan tinggi (atau kustom):</p> <ul style="list-style-type: none"> • Bitrate maks: 900.000 bps • Framerate: 15 fps • Resolusi: 360x640 	<p>Lapisan tinggi (atau kustom):</p> <ul style="list-style-type: none"> • Bitrate maks: 1.700.000 bps • Framerate: 30 fps • Resolusi: 1280x720
<p>Lapisan tengah: tidak ada (tidak diperlukan, karena perbedaan antara bitrate lapisan tinggi dan rendah di ponsel sempit)</p>	<p>Lapisan tengah:</p> <ul style="list-style-type: none"> • Bitrate maks: 700.000 bps • Framerate: 20 fps • Resolusi: 640x360
<p>Lapisan rendah:</p> <ul style="list-style-type: none"> • Bitrate maks: 150.000 bps 	<p>Lapisan rendah:</p> <ul style="list-style-type: none"> • Bitrate maks: 200.000 bps

Seluler (Android, iOS)	Web (Chrome)
<ul style="list-style-type: none">• Framerate: 15 fps• Resolusi: 180x320	<ul style="list-style-type: none">• Framerate: 15 fps• Resolusi: 320x180

Mengkonfigurasi Layered Encoding dengan Simulcast

Untuk menggunakan encoding berlapis dengan simulcast, Anda [harus memilih fitur tersebut, dan mengaktifkannya](#) pada klien. Jika Anda mengaktifkannya, Anda akan melihat peningkatan bitrate keseluruhan yang ditransmisikan, dengan manfaat pembekuan video yang lebih sedikit.

Android

```
// Opt-out of Simulcast
StageVideoConfiguration config = new StageVideoConfiguration();
config.simulcast.setEnabled(true);

ImageLocalStageStream cameraStream = new ImageLocalStageStream(frontCamera, config);

// Other Stage implementation code
```

iOS

```
// Opt-out of Simulcast
let config = IVSLocalStageStreamVideoConfiguration()
config.simulcast.enabled = true

let cameraStream = IVSLocalStageStream(device: camera, configuration: config)

// Other Stage implementation code
```

Web

```
// Opt-out of Simulcast
let cameraStream = new LocalStageStream(cameraDevice, {
  simulcast: { enabled: true }
})

// Other Stage implementation code
```

Konfigurasi Streaming

Bagian ini mengeksplorasi konfigurasi lain yang dapat Anda buat untuk aliran video dan audio Anda.

Mengubah Bitrate Stream Video

Untuk mengubah bitrate streaming video Anda, gunakan contoh konfigurasi berikut.

Android

```
StageVideoConfiguration config = new StageVideoConfiguration();

// Update Max Bitrate to 1.5mbps
config.setMaxBitrate(1500000);

ImageLocalStageStream cameraStream = new ImageLocalStageStream(frontCamera, config);

// Other Stage implementation code
```

iOS

```
let config = IVSLocalStageStreamVideoConfiguration();

// Update Max Bitrate to 1.5mbps
try! config.setMaxBitrate(1500000);

let cameraStream = IVSLocalStageStream(device: camera, configuration: config);

// Other Stage implementation code
```

Web

```
// Note: On web it is also recommended to configure the framerate of your device from
userMedia
const camera = await navigator.mediaDevices.getUserMedia({
  video: {
    bitrate: {
      ideal: 1500,
      max: 1500,
    },
  },
},
```

```
});  
  
let cameraStream = new LocalStageStream(camera.getVideoTracks()[0], {  
  // Update Max Bitrate to 1.5mbps or 1500kbps  
  maxBitrate: 1500  
})  
  
// Other Stage implementation code
```

Mengubah Framerate Stream Video

Untuk mengubah framerate streaming video Anda, gunakan contoh konfigurasi berikut.

Android

```
StageVideoConfiguration config = new StageVideoConfiguration();  
  
// Update target framerate to 10fps  
config.targetFramerate(10);  
  
ImageLocalStageStream cameraStream = new ImageLocalStageStream(frontCamera, config);  
  
// Other Stage implementation code
```

iOS

```
let config = IVSLocalStageStreamVideoConfiguration();  
  
// Update target framerate to 10fps  
try! config.targetFramerate(10);  
  
let cameraStream = IVSLocalStageStream(device: camera, configuration: config);  
  
// Other Stage implementation code
```

Web

```
// Note: On web it is also recommended to configure the framerate of your device from  
userMedia  
const camera = await navigator.mediaDevices.getUserMedia({  
  video: {  
    frameRate: {
```

```
        ideal: 10,  
        max: 10,  
    },  
},  
});  
  
let cameraStream = new LocalStageStream(camera.getVideoTracks()[0], {  
    // Update Max Framerate to 10fps  
    maxFramerate: 10  
})  
// Other Stage implementation code
```

Mengoptimalkan Audio Bitrate dan Dukungan Stereo

Untuk mengubah pengaturan bitrate dan stereo aliran audio Anda, gunakan contoh konfigurasi berikut.

Web

```
// Note: Disable autoGainControl, echoCancellation, and noiseSuppression when enabling  
stereo.  
const camera = await navigator.mediaDevices.getUserMedia({  
    audio: {  
        autoGainControl: false,  
        echoCancellation: false,  
        noiseSuppression: false  
    },  
});  
  
let audioStream = new LocalStageStream(camera.getAudioTracks()[0], {  
    // Optional: Update Max Audio Bitrate to 96Kbps. Default is 64Kbps  
    maxAudioBitrateKbps: 96,  
  
    // Signal stereo support. Note requires dual channel input source.  
    stereo: true  
})  
  
// Other Stage implementation code
```

Android

```
StageAudioConfiguration config = new StageAudioConfiguration();
```

```
// Update Max Bitrate to 96Kbps. Default is 64Kbps.
config.setMaxBitrate(96000);

AudioLocalStageStream microphoneStream = new AudioLocalStageStream(microphone, config);

// Other Stage implementation code
```

iOS

```
let config = IVSLocalStageStreamConfiguration();

// Update Max Bitrate to 96Kbps. Default is 64Kbps.
try! config.audio.setMaxBitrate(96000);

let microphoneStream = IVSLocalStageStream(device: microphone, config: config);

// Other Stage implementation code
```

Optimasi yang Disarankan

Skenario	Rekomendasi
Streaming dengan teks, atau konten yang bergerak lambat, seperti presentasi atau slide	Gunakan encoding berlapis dengan simulcast atau konfigurasi aliran dengan framerate yang lebih rendah.
Streaming dengan aksi atau banyak gerakan	Gunakan encoding berlapis dengan simulcast .
Streaming dengan percakapan atau sedikit gerakan	Gunakan encoding berlapis dengan simulcast atau pilih audio saja (lihat “Berlangganan Peserta” di Panduan SDK Siaran Streaming Real-Time: Web, Android, dan iOS).
Pengguna streaming dengan data terbatas	Gunakan encoding berlapis dengan simulcast atau, jika Anda ingin penggunaan data yang lebih rendah untuk semua orang, konfigurasi framerate yang lebih rendah dan turunkan bitrate secara manual.

Sumber Daya dan Dukungan (Streaming Real-Time)

Sumber daya

<https://ivs.rocks/> adalah situs khusus untuk menelusuri konten yang dipublikasikan (demo, sampel kode, posting blog), perkiraan biaya, dan pengalaman Amazon IVS melalui demo langsung.

Demo



Demo streaming real-time IVS untuk iOS dan Android menunjukkan kepada pengembang cara menggunakan Amazon IVS untuk membangun real-time yang menarik, social-user-generated aplikasi konten. Aplikasi ini memiliki umpan yang dapat digulir dari aliran real-time yang dibuat pengguna. Pengguna dapat membuat streaming video dan ruang khusus audio. Tamu streaming video dapat bergabung dalam mode spot tamu atau versus (VS). Petunjuk tentang cara menerapkan backend yang diperlukan dan membangun aplikasi tersedia di berikut ini GitHub repositori:

- iOS: <https://github.com/aws-samples/amazon-ivs-real-time-for-ios-demo/>

- Android: <https://github.com/aws-samples/amazon-ivs-real-time-for-android-demo/>
- Backend: <https://github.com/aws-samples/amazon-ivs-real-time-serverless-demo/>

Dukungan

Yang [AWS Pusat Dukungan](#) menawarkan berbagai paket yang menyediakan akses ke alat dan keahlian untuk mendukung Anda AWS solusi. Semua paket dukungan menyediakan akses 24/7 ke layanan pelanggan. Untuk dukungan teknis dan lebih banyak sumber daya untuk merencanakan, menyebarkan, dan meningkatkan AWS lingkungan, pilih rencana dukungan yang paling sesuai dengan Anda AWS kasus penggunaan.

[AWS Dukungan Premium](#) adalah one-on-one, saluran dukungan respons cepat untuk membantu Anda membangun dan menjalankan aplikasi AWS.

[AWS Re: posting](#) adalah situs Tanya Jawab berbasis komunitas bagi pengembang untuk mendiskusikan pertanyaan teknis terkait dengan Amazon IVS.

[Hubungi Kami](#) memiliki tautan untuk pertanyaan nonteknis tentang penagihan atau akun Anda. Untuk pertanyaan teknis, gunakan forum diskusi atau tautan dukungan di atas.

Glosarium

Lihat juga [glosarium AWS](#). Pada tabel di bawah ini, LL adalah singkatan dari IVS low-latency streaming; RT, IVS real-time streaming.

Istilah	Deskripsi	LL	RT	Obrolan
AAC	Pengkodean Audio Tingkat Lanjut. AAC adalah standar pengkodean audio untuk kompresi audio digital lossy . Dirancang untuk menjadi penerus format MP3, AAC umumnya mencapai kualitas suara yang lebih tinggi daripada MP3 pada bitrate yang sama. AAC telah distandarisasi oleh ISO dan IEC sebagai bagian dari spesifikasi MPEG-2 dan MPEG-4.	✓	✓	
Streaming bitrate adaptif	Streaming Adaptive Bitrate (ABR) memungkinkan pemutar IVS untuk beralih ke bitrate yang lebih rendah ketika kualitas koneksi menurun, dan untuk beralih kembali ke bitrate yang lebih tinggi ketika kualitas koneksi meningkat.	✓		
Streaming adaptif	Lihat Layered encoding dengan simulcast .		✓	
Pengguna administratif	Pengguna AWS dengan akses administratif ke sumber daya dan layanan yang tersedia di akun AWS. Lihat Terminologi dalam Panduan Pengguna AWS Setup.	✓	✓	✓
ARN	Nama Sumber Daya Amazon , pengenal unik untuk sumber daya AWS. Format ARN spesifik tergantung pada jenis sumber daya. Untuk format ARN yang digunakan oleh sumber daya IVS, lihat di Referensi Otorisasi Layanan.	✓	✓	✓

Istilah	Deskripsi	LL	RT	Obrolan
Rasio aspek	Menjelaskan rasio lebar bingkai dengan tinggi bingkai. Misalnya, 16:9 adalah rasio aspek yang sesuai dengan resolusi Full HD atau 1080p.	✓	✓	
Modus audio	Konfigurasi audio preset atau kustom yang dioptimalkan untuk berbagai jenis pengguna perangkat seluler dan peralatan yang mereka gunakan. Lihat SDK Siaran IVS: Mode Audio Seluler (Streaming Waktu Nyata) .		✓	
AVC, H.264, MPEG-4 Bagian 10	Advanced Video Coding, juga disebut sebagai H.264 atau MPEG-4 Part 10, standar kompresi video untuk kompresi video digital lossy.	✓	✓	
Penggantian latar belakang	Jenis filter kamera yang memungkinkan pembuat live-stream mengubah latar belakang mereka. Lihat Penggantian Latar Belakang di SDK Siaran IVS: Filter Kamera Pihak Ketiga (Streaming Waktu Nyata).		✓	
Bitrate	Metrik streaming untuk jumlah bit yang ditransmisikan atau diterima per detik.	✓	✓	
Siaran, penyiar	Istilah lain untuk streaming , streamer .	✓		
Penyangga	Suatu kondisi yang terjadi ketika perangkat pemutaran tidak dapat mengunduh konten sebelum konten seharusnya diputar. Buffering dapat bermanifestasi dalam beberapa cara: konten dapat berhenti dan mulai secara acak (juga dikenal sebagai gagap), konten dapat berhenti untuk jangka waktu yang lama (juga dikenal sebagai pembekuan), atau pemutar IVS dapat menjeda pemutaran.	✓	✓	

Istilah	Deskripsi	LL	RT	Obrolan
Daftar putar rentang byte	<p>Daftar putar yang lebih terperinci daripada daftar putar HLS standar. Daftar putar HLS standar terdiri dari file media 10 detik. Dengan playlist byte-range, durasi segmen sama dengan interval keyframe yang dikonfigurasi untuk streaming.</p> <p>Daftar putar rentang byte hanya tersedia untuk siaran yang direkam secara otomatis ke bucket S3. Itu dibuat selain daftar putar HLS. Lihat Daftar Putar Byte-Range dalam Rekam Otomatis ke Amazon S3 (Streaming Latensi Rendah).</p>	✓		
CBR	<p>Bitrate Konstan, metode kontrol kecepatan untuk encoder yang mempertahankan bitrate yang konsisten di seluruh pemutaran video, terlepas dari apa yang terjadi selama siaran. Tidurkan dalam aksi dapat dilapisi untuk mencapai bitrate yang diinginkan, dan puncak dapat dikuantisasi dengan menyesuaikan kualitas pengkodean agar sesuai dengan bitrate target. Kami sangat menyarankan menggunakan CBR alih-alih VBR.</p>	✓	✓	
CDN	<p>Jaringan Pengiriman Konten atau Jaringan Distribusi Konten, solusi yang didistribusikan secara geografis yang mengoptimalkan pengiriman konten seperti streaming video dengan membawanya lebih dekat ke tempat pengguna berada.</p>	✓		

Istilah	Deskripsi	LL	RT	Obrolan
Channel	Sumber daya IVS yang menyimpan konfigurasi untuk streaming, termasuk server ingest , kunci aliran , URL pemutaran , dan opsi perekaman. Streamer menggunakan tombol streaming yang terkait dengan saluran untuk memulai siaran. Semua metrik dan peristiwa yang dihasilkan selama siaran dikaitkan dengan sumber daya saluran.	✓		
Jenis saluran	Menentukan resolusi dan frame rate yang diijinkan untuk saluran . Lihat Jenis Saluran di Referensi API Streaming Latensi Rendah IVS .	✓		
Pencatatan obrolan	Opsi lanjutan yang dapat diaktifkan dengan mengaitkan konfigurasi logging dengan ruang obrolan .			✓
Ruang obrolan	Sumber daya IVS yang menyimpan konfigurasi untuk sesi obrolan, termasuk fitur opsional seperti Message Review Handler dan Chat Logging . Lihat Langkah 2: Buat Ruang Obrolan di Memulai Obrolan IVS.			✓
Komposisi sisi klien	Menggunakan perangkat host untuk mencampur aliran audio dan video dari peserta panggung dan kemudian mengirimkannya sebagai aliran komposit ke saluran IVS . Hal ini memungkinkan kontrol lebih besar atas tampilan komposisi dengan biaya pemanfaatan sumber daya klien yang lebih tinggi dan risiko panggung yang lebih tinggi atau masalah host yang berdampak pada pemirsa. Lihat juga komposisi sisi server .	✓	✓	
CloudFront	Layanan CDN yang disediakan oleh Amazon.	✓		

Istilah	Deskripsi	LL	RT	Obrolan
CloudTrail	Layanan AWS untuk mengumpulkan, memantau, menganalisis, dan mempertahankan peristiwa dan aktivitas akun dari AWS dan sumber eksternal. Lihat Mencatat Panggilan API IVS dengan AWS CloudTrail .	✓	✓	✓
CloudWatch	Layanan AWS untuk memantau aplikasi, menanggapi perubahan kinerja, mengoptimalkan penggunaan sumber daya, dan memberikan wawasan tentang kesehatan operasional. Anda dapat menggunakannya CloudWatch untuk memantau metrik IVS; lihat Memantau Streaming Waktu Nyata IVS dan Pemantauan Streaming Latensi Rendah IVS.	✓	✓	✓
Komposisi	Proses menggabungkan aliran audio dan video dari berbagai sumber menjadi satu aliran.	✓	✓	
Pipa komposisi	Urutan langkah-langkah pemrosesan yang diperlukan untuk menggabungkan beberapa aliran dan menyandikan aliran yang dihasilkan.	✓	✓	
Kompresi	Pengkodean informasi menggunakan bit lebih sedikit dari representasi asli. Kompresi tertentu adalah lossless atau lossy. Kompresi lossless mengurangi bit dengan mengidentifikasi dan menghilangkan redundansi statistik. Tidak ada informasi yang hilang dalam kompresi lossless. Kompresi lossy mengurangi bit dengan menghapus informasi yang tidak perlu atau kurang penting.	✓	✓	
Bidang kontrol	Menyimpan informasi tentang sumber daya IVS seperti saluran , tahap , atau ruang obrolan dan menyediakan antarmuka untuk membuat dan mengelola sumber daya ini. Ini bersifat regional (berdasarkan wilayah AWS).	✓	✓	✓

Istilah	Deskripsi	LL	RT	Obrolan
CORS	Cross-Origin Resource Sharing, fitur AWS yang memungkinkan aplikasi web klien yang dimuat dalam satu domain untuk berinteraksi dengan sumber daya seperti bucket S3 di domain yang berbeda. Akses dapat dikonfigurasi berdasarkan header, metode HTTP, dan domain asal. Lihat Menggunakan berbagi sumber daya lintas asal (CORS) - Layanan Penyimpanan Sederhana Amazon di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon .	✓		
Sumber gambar khusus	Antarmuka yang disediakan oleh IVS Broadcast SDK yang memungkinkan aplikasi memberikan input gambarnya sendiri alih-alih terbatas pada kamera preset.	✓	✓	
Bidang data	Infrastruktur yang membawa data dari ingest ke egress. Ini beroperasi berdasarkan konfigurasi yang dikelola di bidang kontrol dan tidak terbatas pada wilayah AWS.	✓	✓	✓
Encoder, pengkodean	Proses konversi konten video dan audio ke dalam format digital, cocok untuk streaming. Pengkodean dapat berbasis perangkat keras atau perangkat lunak.	✓	✓	
Peristiwa	Pemberitahuan otomatis yang diterbitkan oleh IVS ke layanan AmazonEventBridge pemantauan. Suatu peristiwa mewakili perubahan keadaan atau kesehatan dari sumber daya streaming seperti panggung atau pipa komposisi . Lihat Menggunakan Amazon EventBridge dengan Streaming Latensi Rendah IVS dan Menggunakan Amazon EventBridge dengan Streaming Waktu Nyata IVS .	✓	✓	✓

Istilah	Deskripsi	LL	RT	Obrolan
FFmpeg	Proyek perangkat lunak sumber terbuka dan gratis yang terdiri dari serangkaian perpustakaan dan program untuk menangani file dan aliran video dan audio. FFmpeg menyediakan solusi lintas platform untuk merekam, mengonversi, dan mengalirkan audio dan video.	✓		
Aliran terfragmentasi	Dibuat ketika siaran terputus dan kemudian terhubung kembali dalam interval yang ditentukan dalam konfigurasi perekaman saluran . Beberapa aliran yang dihasilkan dianggap sebagai siaran tunggal dan digabungkan menjadi satu aliran rekaman. Lihat Menggabungkan Streaming Terfragmentasi dalam Rekam Otomatis ke Amazon S3 (Streaming Latensi Rendah).	✓		
Frame rate	Metrik streaming untuk jumlah frame video yang dikirim atau diterima per detik.	✓	✓	
HLS	HTTP Live Streaming (HLS), protokol komunikasi streaming bitrate adaptif berbasis HTTP yang digunakan untuk mengirimkan aliran IVS kepada pemirsa.	✓		
Daftar putar HLS	Daftar segmen media yang membentuk aliran. Daftar putar HLS standar terdiri dari file media 10 detik. HLS juga mendukung lebih banyak daftar putar byte-range granular.	✓		
Host	Peserta acara real-time yang mengirimkan video dan/atau audio ke panggung.		✓	
IAM	Identity and Access Management, layanan AWS yang memungkinkan pengguna mengelola identitas dan akses ke layanan dan sumber daya AWS dengan aman, termasuk IVS.	✓	✓	✓

Istilah	Deskripsi	LL	RT	Obrolan
Tertelan	Proses IVS untuk menerima aliran video dari host atau penyiar untuk diproses atau dikirim ke pemirsa atau peserta lain.	✓	✓	
Server tertelan	<p>Menerima streaming video dan mengirimkannya ke sistem transcoding, di mana aliran ditransmuxed atau ditranskode ke HLS untuk dikirim ke pemirsa.</p> <p>Server Ingest adalah komponen IVS spesifik yang menerima aliran untuk saluran, bersama dengan protokol konsumsi (RTMP, RTMPS). Lihat informasi tentang membuat saluran di Memulai dengan Streaming Latensi Rendah IVS.</p>		✓	
Video yang saling terkait	Mentransmisikan dan menampilkan hanya garis ganjil atau genap dari frame berikutnya untuk menciptakan penggandaan frame rate yang dirasakan tanpa menghabiskan bandwidth ekstra. Kami tidak menyarankan menggunakan video interlaced karena masalah kualitas video.	✓	✓	
JSON	JavaScript Object Notation, format file standar terbuka yang menggunakan teks yang dapat dibaca manusia untuk mengirimkan objek data yang terdiri dari pasangan nilai atribut dan tipe data array atau nilai serialisasi lainnya.	✓	✓	✓
Bingkai kunci, bingkai delta, interval keyframe	Keyframe (juga disebut sebagai intra-coded atau i-frame) adalah bingkai penuh gambar dalam video. Frame berikutnya, delta frame (juga disebut sebagai prediksi atau p-frame), hanya berisi informasi yang telah berubah. Keyframe akan muncul beberapa kali dalam aliran , tergantung pada interval keyframe yang ditentukan dalam encoder.	✓	✓	

Istilah	Deskripsi	LL	RT	Obrolan
Lambda	Layanan AWS untuk menjalankan kode (disebut sebagai fungsi Lambda) tanpa menyediakan infrastruktur server apa pun. Fungsi Lambda dapat berjalan sebagai respons terhadap peristiwa dan permintaan pemanggilan, atau berdasarkan jadwal. Misalnya, Obrolan IVS menggunakan fungsi Lambda untuk mengaktifkan peninjauan pesan untuk ruang obrolan.	✓	✓	✓
Latensi, glass-to-glass latensi	Penundaan transfer data. IVS mendefinisikan rentang latensi sebagai: <ul style="list-style-type: none"> • Latensi rendah: di bawah 3 detik • Latensi waktu nyata: di bawah 300 ms <p>lass-to-glassLatensi G mengacu pada penundaan dari saat kamera menangkap streaming langsung hingga saat streaming muncul di layar pemirsa.</p>	✓	✓	
Pengkodean berlapis dengan simulcast	Memungkinkan pengkodean simultan dan penerbitan beberapa aliran video dengan tingkat kualitas yang berbeda. Lihat Streaming Adaptif: Pengkodean Berlapis dengan Simulcast dalam Optimasi Streaming Waktu Nyata.		✓	
Penangan ulasan pesan	Memungkinkan pelanggan IVS Chat untuk secara otomatis meninjau/memfilter pesan obrolan pengguna sebelum dikirim ke ruang obrolan. Ini diaktifkan dengan mengaitkan fungsi Lambda dengan ruang obrolan. Lihat Membuat Fungsi Lambda di Penangan Tinjauan Pesan Obrolan.			✓

Istilah	Deskripsi	LL	RT	Obrolan
Mixer	<p>Fitur IVS Mobile Broadcast SDK yang mengambil beberapa sumber audio dan video dan menghasilkan satu output. Ini mendukung pengelolaan elemen video dan audio di layar yang mewakili sumber seperti kamera, mikrofon, tangkapan layar, dan audio dan video yang dihasilkan oleh aplikasi. Output kemudian dapat dialirkan ke IVS. Lihat Mengonfigurasi Sesi Siaran untuk Pencampuran dalam SDK Siaran IVS: Panduan Mixer (Streaming Latensi Rendah).</p>	✓		
Streaming multi-host	<p>Menggabungkan aliran dari beberapa host ke dalam satu aliran. Ini dapat dicapai dengan menggunakan komposisi sisi klien atau sisi server.</p> <p>Streaming multi-host memungkinkan skenario seperti mengundang pemirsa ke panggung untuk Tanya Jawab, kompetisi antar host, obrolan video, dan host berbicara satu sama lain di depan khalayak yang besar.</p>		✓	
Daftar putar multivariant	<p>Indeks semua aliran varian yang tersedia untuk siaran.</p>	✓		
OAC	<p>Origin Access Control, mekanisme untuk membatasi akses ke bucket S3, sehingga konten seperti aliran rekaman hanya dapat dilayani melalui CloudFrontCDN.</p>	✓		

Istilah	Deskripsi	LL	RT	Obrolan
OBS	Open Broadcaster Software, perangkat lunak gratis dan open source untuk perekaman video dan streaming langsung. OBS menawarkan alternatif (untuk IVS broadcast SDK) untuk penerbitan desktop. Streamer yang lebih canggih yang akrab dengan OBS mungkin lebih menyukainya karena fitur produksinya yang canggih, seperti transisi adegan, pencampuran audio, dan grafik overlay.	✓	✓	
Peserta	Pengguna real-time yang terhubung ke panggung sebagai host atau penampil .		✓	
Token peserta	Mengautentikasi peserta acara real-time saat mereka bergabung dengan panggung . Token peserta juga mengontrol apakah peserta dapat mengirim video ke panggung.		✓	
Token pemutaran, playback key pair	Mekanisme otorisasi yang memungkinkan pelanggan membatasi pemutaran video di saluran pribadi . Token pemutaran dihasilkan dari playback key pair. Playback key pair adalah sepasang kunci publik-pribadi yang digunakan untuk menandatangani dan memvalidasi token otorisasi penampil untuk pemutaran. Lihat Membuat atau Mengimpor Kunci Pemutaran dalam Menyiapkan Saluran Pribadi dan lihat titik akhir Playback Key Pair di Referensi API Latensi Rendah IVS .	✓		

Istilah	Deskripsi	LL	RT	Obrolan
URL pemutaran	Mengidentifikasi alamat yang digunakan penampil untuk memulai pemutaran saluran tertentu. Alamat ini dapat digunakan secara global. IVS secara otomatis memilih lokasi terbaik di jaringan pengiriman konten global IVS untuk mengirimkan video ke setiap pemirsa . Lihat informasi tentang membuat saluran di Memulai dengan Streaming Latensi Rendah IVS .	✓		
Saluran pribadi	Memungkinkan pelanggan membatasi akses ke aliran mereka menggunakan mekanisme otorisasi berdasarkan token pemutaran . Lihat Alur Kerja untuk Saluran Pribadi di Menyiapkan Saluran Pribadi.	✓		
Video progresif	Mentransmisikan dan menampilkan semua baris dari setiap frame secara berurutan. Kami merekomendasikan penggunaan video progresif selama semua tahap siaran.	✓	✓	
Kuota	Jumlah maksimum sumber daya layanan IVS atau operasi untuk akun AWS Anda. Artinya, batas ini per akun AWS, kecuali dinyatakan lain. Semua kuota diberlakukan per wilayah. Lihat titik akhir dan kuota Layanan Video Interaktif Amazon di Panduan Referensi Umum AWS.	✓	✓	✓

Istilah	Deskripsi	LL	RT	Obrolan
Wilayah	<p>Menyediakan akses ke layanan AWS yang secara fisik berada di wilayah geografis tertentu. Wilayah memberikan toleransi kesalahan, stabilitas, dan ketahanan, dan juga dapat mengurangi latensi. Dengan Wilayah, Anda dapat membuat sumber daya redundan yang tetap tersedia dan tidak terpengaruh oleh pemadaman regional.</p> <p>Sebagian besar permintaan layanan AWS dikaitkan dengan wilayah geografis tertentu. Sumber daya yang Anda buat di satu wilayah tidak ada di wilayah lain kecuali Anda secara eksplisit menggunakan fitur replikasi yang ditawarkan oleh layanan AWS. Misalnya, Amazon S3 mendukung replikasi lintas wilayah. Beberapa layanan, seperti IAM, tidak memiliki sumber daya lintas wilayah.</p>	✓	✓	✓
Penyelesaian	Menjelaskan jumlah piksel dalam satu bingkai video, misalnya, Full HD atau 1080p mendefinisikan bingkai dengan 1920x1080 piksel.	✓	✓	
Pengguna root	Pemilik akun AWS. Pengguna root memiliki akses lengkap ke semua layanan dan sumber daya AWS di akun AWS.	✓	✓	✓
RTMP, RTMPS	Real-Time Messaging Protocol, standar industri untuk mentransmisikan audio, video, dan data melalui jaringan. RTMPS adalah versi aman dari RTMP, berjalan melalui koneksi Transport Layer Security (TLS/SSL).	✓	✓	
Bucket S3	Kumpulan objek yang disimpan di Amazon S3. Banyak kebijakan, termasuk akses dan replikasi, ditentukan pada tingkat bucket dan berlaku untuk semua objek di bucket. Misalnya, siaran IVS disimpan sebagai beberapa objek dalam ember S3.	✓		

Istilah	Deskripsi	LL	RT	Obrolan
SDK	Software Development Kit, kumpulan perpustakaan untuk pengembang membangun aplikasi dengan IVS.	✓	✓	✓
Segmentasi selfie	Memungkinkan penggantian latar belakang dalam streaming langsung, menggunakan solusi khusus klien yang menerima gambar kamera sebagai input dan mengembalikan topeng yang memberikan skor kepercayaan untuk setiap piksel gambar, yang menunjukkan apakah itu di latar depan atau latar belakang. Lihat Penggantian Latar Belakang di SDK Siaran IVS: Filter Kamera Pihak Ketiga (Streaming Waktu Nyata).		✓	
Versi semantik	Format versi dalam bentuk Major.Minor.Patch. Perbaikan bug yang tidak memengaruhi kenaikan API versi tambalan, penambahan/perubahan API yang kompatibel ke belakang meningkatkan versi minor, dan perubahan API yang tidak kompatibel ke belakang meningkatkan versi utama.	✓	✓	✓
Komposisi sisi server	Menggunakan server IVS untuk mencampur audio dan video dari peserta panggung dan kemudian mengirimkan video campuran ini ke saluran IVS untuk menjangkau audiens yang lebih besar atau menyimpannya dalam ember S3 . Komposisi sisi server mengurangi beban klien, meningkatkan ketahanan siaran, dan memungkinkan penggunaan bandwidth yang lebih efisien. Lihat juga komposisi sisi klien .		✓	

Istilah	Deskripsi	LL	RT	Obrolan
Kuota layanan	Layanan AWS yang membantu Anda mengelola kuota untuk banyak layanan AWS dari satu lokasi. Selain mencari nilai kuota, Anda juga dapat meminta peningkatan kuota dari konsol Service Quotas.	✓	✓	✓
Peran terkait layanan	Jenis peran IAM unik yang ditautkan langsung ke layanan AWS. Peran terkait layanan dibuat secara otomatis oleh IVS dan menyertakan semua izin yang diperlukan layanan untuk memanggil layanan AWS lain atas nama Anda, misalnya, untuk mengakses bucket S3. Lihat Menggunakan Peran Tertaut Layanan untuk IVS di Keamanan IVS .	✓		
Tahap	Sumber daya IVS yang mewakili ruang virtual tempat peserta acara real-time dapat bertukar video secara real time. Lihat Membuat Panggung dalam Memulai Streaming Waktu Nyata IVS .		✓	
Sesi panggung	Dimulai ketika peserta pertama bergabung dengan panggung dan berakhir beberapa menit setelah peserta terakhir berhenti mempublikasikan ke panggung. Tahap yang berumur panjang mungkin memiliki beberapa sesi selama masa hidupnya.		✓	
Aliran	Data yang mewakili konten video atau audio dikirim terus menerus dari sumber ke tujuan.	✓	✓	
Kunci aliran	Pengidentifikasi yang ditetapkan oleh IVS saat Anda membuat saluran ; itu digunakan untuk mengotorisasi streaming ke saluran. Perlakukan tombol streaming seperti rahasia, karena siapa pun yang memilikinya dapat melakukan streaming ke saluran. Lihat Memulai Streaming Latensi Rendah IVS .	✓		

Istilah	Deskripsi	LL	RT	Obrolan
Streaming kelaparan	<p>Penundaan atau penghentian pengiriman aliran ke IVS. Ini terjadi ketika IVS tidak menerima jumlah bit yang diharapkan bahwa perangkat pengkodean yang diiklankan akan dikirim selama jangka waktu tertentu. Terjadinya kelaparan aliran menghasilkan peristiwa kelaparan aliran.</p> <p>Dari sudut pandang pemirsa, kelaparan streaming dapat muncul sebagai video yang tertinggal, buffer, atau macet. Kelaparan aliran bisa singkat (kurang dari 5 detik) atau panjang (beberapa menit), tergantung pada situasi spesifik yang mengakibatkan kelaparan aliran. Lihat Apa itu Stream Kelaparan di FAQ Pemecahan Masalah.</p>	✓	✓	
Streamer	Seseorang atau perangkat yang mengirim aliran video atau audio ke IVS.	✓	✓	
Pelanggan	Peserta acara real-time yang menerima video dan/atau audio dari tuan rumah. Lihat Apa itu Streaming Waktu Nyata IVS.		✓	
Tag	Label metadata yang Anda tetapkan ke sumber daya AWS. Tag dapat membantu Anda mengidentifikasi dan mengatur sumber daya AWS Anda. Pada halaman landing dokumentasi IVS , lihat “Penandaan” di dokumentasi API IVS mana pun (untuk streaming real-time, streaming latensi rendah, atau obrolan).	✓	✓	✓

Istilah	Deskripsi	LL	RT	Obrolan
Filter kamera pihak ketiga	Komponen perangkat lunak yang dapat diintegrasikan dengan IVS Broadcast SDK untuk memungkinkan aplikasi memproses gambar sebelum memberikannya ke Broadcast SDK sebagai sumber gambar khusus . Filter kamera pihak ketiga dapat memproses gambar dari kamera, menerapkan efek filter, dll.	✓	✓	
Thumbnail	Gambar berukuran kecil yang diambil dari aliran. Secara default, thumbnail dihasilkan setiap 60 detik, tetapi interval yang lebih pendek dapat dikonfigurasi. Resolusi thumbnail tergantung pada jenis saluran . Lihat Merekam Konten dalam Rekam Otomatis ke Amazon S3 (Streaming Latensi Rendah).	✓		
Metadata berjangka waktu	<p>Metadata terkait dengan stempel waktu tertentu dalam aliran. Ini dapat ditambahkan secara terprogram menggunakan API IVS dan menjadi terkait dengan frame tertentu. Ini memastikan bahwa semua pemirsa menerima metadata pada titik yang sama relatif terhadap aliran.</p> <p>Metadata berwaktu dapat digunakan untuk memicu tindakan pada klien seperti memperbarui statistik tim selama acara olahraga. Lihat Menyematkan Metadata dalam Stream Video.</p>	✓		
Transcoding	Mengkonversi video dan audio dari satu format ke format lainnya. Aliran masuk dapat ditranskode ke format yang berbeda pada beberapa bitrate dan resolusi, untuk mendukung berbagai perangkat pemutaran dan kondisi jaringan.	✓	✓	

Istilah	Deskripsi	LL	RT	Obrolan
Transmuxing	<p>Pengemasan ulang sederhana dari aliran yang dicerna ke IVS, tanpa pengkodean ulang aliran video. “Transmux” adalah kependekan dari transcode multiplexing, sebuah proses yang mengubah format file audio dan/atau video sambil menyimpan beberapa atau semua aliran asli. Transmuxing mengkonversi ke format wadah yang berbeda tanpa mengubah isi file. Dibedakan dari transcoding.</p>	✓	✓	
Aliran varian	<p>Satu set pengkodean siaran yang sama dalam beberapa tingkat kualitas yang berbeda. Setiap aliran varian dikodekan sebagai daftar putar HLS terpisah. Indeks aliran varian yang tersedia disebut sebagai daftar putar multivarian.</p> <p>Setelah pemutar IVS menerima daftar putar multivarian dari IVS, ia kemudian dapat memilih antara aliran varian selama pemutaran, berubah bolak-balik dengan mulus saat kondisi jaringan berubah.</p>	✓		
VBR	<p>Variable Bitrate, metode rate-control untuk encoder yang menggunakan bitrate dinamis yang berubah sepanjang pemutaran, tergantung pada tingkat detail yang diperlukan. Kami sangat menyarankan an agar tidak menggunakan VBR karena masalah kualitas video; gunakan CBR sebagai gantinya.</p>	✓	✓	

Istilah	Deskripsi	LL	RT	Obrolan
Lihat	<p>Sesi menonton unik yang secara aktif mengunduh atau memutar video. Tampilan adalah dasar untuk kuota tampilan bersamaan.</p> <p>Tampilan dimulai saat sesi menonton dimulai pemutaran video. Tampilan berakhir ketika sesi menonton menghentikan pemutaran video. Pemutaran adalah satu-satunya indikator pemirsa; heuristik keterlibatan seperti level audio, fokus tab browser, dan kualitas video tidak dipertimbangkan. Saat menghitung penayangan, IVS tidak mempertimbangkan legitimasi pemirsa individu atau mencoba menghapus duplikasi pemirsa lokal, seperti beberapa pemutar video pada satu mesin. Lihat Kuota Lainnya di Service Quotas (Streaming Latensi Rendah).</p>	✓		
Orang yang melihat	Seseorang menerima aliran dari IVS.	✓		
WebRTC	<p>Web Real-Time Communication, sebuah proyek sumber terbuka yang menyediakan browser web dan aplikasi seluler dengan komunikasi real-time. Ini memungkinkan komunikasi audio dan video untuk bekerja di dalam halaman web dengan memungkinkan peer-to-peer komunikasi langsung, menghilangkan kebutuhan untuk menginstal plugin atau mengunduh aplikasi asli.</p> <p>Teknologi di balik WebRTC diimplementasikan sebagai standar web terbuka dan tersedia sebagai API JavaScript reguler di semua browser utama atau sebagai pustaka untuk klien asli, seperti Android dan iOS.</p>	✓	✓	

Istilah	Deskripsi	LL	RT	Obrolan
CAMBUK	<p>WebRTC-HTTP Ingestion Protocol, protokol berbasis HTTP yang memungkinkan konsumsi konten berbasis WebRTC ke dalam layanan streaming dan/atau CDN. WHIP adalah draf IETF yang dikembangkan untuk membakukan konsumsi WebRTC.</p> <p>WHIP memungkinkan kompatibilitas dengan perangkat lunak seperti OBS, menawarkan alternatif (ke SDK siaran IVS) untuk penerbitan desktop. Streamer yang lebih canggih yang akrab dengan OBS mungkin lebih menyukainya karena fitur produksinya yang canggih, seperti transisi adegan, pencampuran audio, dan grafik overlay</p> <p>WHIP juga bermanfaat dalam situasi di mana menggunakan SDK siaran IVS tidak layak atau disukai. Misalnya, dalam pengaturan yang melibatkan encoder perangkat keras, SDK siaran IVS mungkin bukan pilihan. Namun, jika encoder mendukung WHIP, Anda masih dapat mempublikasikan langsung dari encoder ke IVS.</p> <p>Lihat OBS dan WHIP Support.</p>		✓	
WSS	<p>WebSocket Secure, protokol untuk membangun WebSockets melalui koneksi TLS terenkripsi. Ini digunakan untuk menghubungkan ke titik akhir Obrolan IVS. Lihat Langkah 4: Kirim dan Terima Pesan Pertama Anda di Memulai Obrolan IVS.</p>			✓

Riwayat Dokumen (Streaming Waktu Nyata)

Perubahan Panduan Pengguna Streaming Waktu Nyata

Perubahan	Deskripsi	Tanggal
OBS dan WHIP Support	Ditambahkan halaman baru. Dokumen ini menjelaskan cara menggunakan encoder yang kompatibel dengan Whip seperti OBS untuk mempublikasikan ke streaming real-time IVS. WHIP (WebRTC-HTTP Ingestion Protocol) adalah draf IETF yang dikembangkan untuk membakukan konsumsi WebRTC.	Februari 6, 2024
Siaran SDK: Android 1.14.1, iOS 1.14.1, Web 1.8.0	Nomor versi dan tautan artefak yang diperbarui untuk rilis baru, dalam panduan SDK real-time-streaming siaran: Android, iOS, dan Web. Pada halaman arahan dokumentasi Amazon IVS , memperbarui tautan Referensi SDK siaran untuk menunjuk ke versi baru. Lihat juga Catatan Rilis Amazon IVS untuk rilis ini. Untuk Panduan Android, kami menambahkan Masalah Diketahui baru (ukuran video kurang dari 176x176).	Februari 1, 2024

Untuk Panduan Web, kami menambahkan Masalah Baru yang Diketahui. Solusinya adalah membatasi resolusi video ke 720p saat memanggil `getUserMedia` atau `getDisplayMedia`.

Dalam Optimasi Streaming Waktu Nyata, kami memperbarui [Konfigurasi Layered Encoding dengan Simulcast](#); sekarang ini dinonaktifkan secara default.

[Siaran SDK: Android 1.13.4, iOS 1.13.4, Web 1.7.0](#)

[Nomor versi dan tautan artefak yang diperbarui untuk rilis baru, dalam panduan SDK real-time-streaming siaran: Android, iOS, dan Web.](#) Pada [halaman arahan dokumentasi Amazon IVS](#), memperbarui tautan Referensi SDK siaran untuk menunjuk ke versi baru. Lihat juga [Catatan Rilis Amazon IVS](#) untuk rilis ini.

Januari 3, 2024

[IVS Glosarium](#)

Memperpanjang glosarium, mencakup istilah IVS real-time, latensi rendah, dan obrolan.

Desember 20, 2023

[Stage Health: CloudWatch Metrik Baru](#)

Mengganti nama metrik PacketLoss (Tahap) menjadi DownloadPacketLoss (Tahap) dan merilis CloudWatch metrik tambahan untuk streaming real-time IVS:

Desember 7, 2023

- DownloadPacketLoss (Panggung, Peserta)
- DroppedFrames (Panggung, Peserta)
- SubscribeBitrate (Panggung, Peserta, MediaType)

Lihat [Memantau Streaming Waktu Nyata IVS](#).

[Kebijakan yang dikelola IAM](#)

Menambahkan dua kebijakan terkelola, IVS ReadOnlyAccess dan FullAccess IVS. Lihat:

5 Desember 2023

- Bagian baru tentang [Kebijakan Terkelola untuk Amazon IVS](#) di halaman Keamanan.
- Perubahan pada [Langkah 3: Siapkan Izin IAM](#) untuk Memulai Streaming Latensi Rendah IVS.

[Siaran SDK: Android 1.13.2, iOS 1.13.2](#)

[Nomor versi dan tautan artefak yang diperbarui untuk rilis baru, dalam panduan SDK real-time-streaming siaran: Android dan iOS.](#)

Desember 4, 2023

Pada [halaman arahan dokumentasi Amazon IVS](#), memperbarui tautan Referensi SDK siaran untuk menunjuk ke versi baru.

Lihat juga [Catatan Rilis](#) Amazon IVS untuk rilis ini.

[Siaran SDK: Android 1.13.1](#)

[Nomor versi dan tautan artefak yang diperbarui untuk rilis baru, dalam panduan SDK real-time-streaming siaran: Android.](#)

21 November 2023

Pada [halaman arahan dokumentasi Amazon IVS](#), memperbarui tautan Referensi SDK siaran untuk menunjuk ke versi baru.

Lihat juga [Catatan Rilis](#) Amazon IVS untuk rilis ini.

[Service Quotas](#)

Mengubah “Resolusi publikasi peserta” dari 1080p menjadi 720p.

November 18, 2023

[Siaran SDK: Android 1.13.0, iOS 1.13.0](#)

[Nomor versi dan tautan artefak yang diperbarui untuk rilis baru, dalam panduan SDK real-time-streaming siaran: Android dan iOS.](#)

17 November 2023

Pada [halaman arahan dokumentasi Amazon IVS](#), memperbarui tautan Referensi SDK siaran untuk menunjuk ke versi baru.

Lihat juga [Catatan Rilis](#) Amazon IVS untuk rilis ini.

Kami juga membuat berbagai pembaruan untuk [Pengoptimalan Streaming](#). Antara lain, fitur “Adaptive Streaming : Layered Encoding with Simulcast” sekarang memerlukan opt-in eksplisit dan hanya didukung dalam versi SDK terbaru.

Rekaman Komposit

Membuat perubahan berikut:

16 November 2023

- Menambahkan halaman [Rekaman Komposit](#) untuk fitur baru ini.
- Memperbarui [Memulai dengan Streaming Waktu Nyata IVS](#) dengan titik akhir S3 dalam kebijakan di “Siapkan Izin IAM.”
- [Service Quotas yang diperbarui dengan kuota call-rate](#) untuk titik akhir baru.

Komposisi sisi server (SSC)

16 November 2023

Komposisi sisi server IVS memungkinkan klien untuk menurunkan komposisi dan penyiaran tahap IVS ke layanan yang dikelola IVS. Siaran SSC dan RTMP ke saluran dipanggil melalui titik akhir bidang kontrol IVS di wilayah asal panggung. Lihat:

- [Memulai](#) — Kami menambahkan titik akhir SSC ke kebijakan di “Siapkan Izin IAM.”
- [Menggunakan Amazon EventBridge dengan IVS](#) — Kami menambahkan metrik baru.
- [Komposisi Sisi Server](#) - Dokumen baru ini mencakup ikhtisar dan instruksi penyiapan.
- [Service Quotas](#) - Kami menambahkan batas call-rate baru dan kuota lainnya.

Lihat juga:

- Perubahan yang tercantum di bawah ini dalam [Perubahan Referensi API Streaming Waktu Nyata IVS](#).
- Perubahan yang tercantum dalam [Riwayat Dokumen](#)

[\(Streaming Latensi Rendah\).](#)

SDK siaran IVS	Dalam ikhtisar Broadcast SDK , kami memperbarui Persyaratan Platform > Platform Asli untuk memperjelas versi SDK mana yang didukung dan kami menambahkan “Browser Seluler (iOS dan Android).”	9 November 2023
	Dalam Panduan Web Siaran , kami menambahkan “Batasan Web Seluler.”	
SDK siaran IVS	Kami menambahkan halaman baru pada Filter Kamera Pihak Ketiga .	9 November 2023
Memulai dengan IVS Real-Time Streaming	Kami memperbarui prosedur di Mengatur Izin IAM .	20 Oktober 2023
Memantau Streaming Waktu Nyata	Di CloudWatch Metrik: Streaming Waktu Nyata IVS , kami menambahkan nilai sampel untuk dimensi.	17 Oktober 2023
Broadcast SDK: Panduan Web	Kami membuat beberapa perubahan pada Memantau Status Bisu Media Peserta Jarak Jauh .	17 Oktober 2023

[Siaran SDK: Web 1.6.0](#)

[Nomor versi dan tautan artefak yang diperbarui untuk rilis baru, dalam panduan SDK real-time-streaming siaran: Web.](#)

16 Oktober 2023

[Halaman arahan dokumentasi Amazon IVS menunjuk ke versi Referensi SDK Siaran saat ini.](#)

Lihat juga [Catatan Rilis Amazon IVS](#) untuk rilis ini.

Dalam Panduan Web, di “Ambil MediaStream dari Perangkat,” kami juga menghapus dua max baris; praktik terbaik adalah menentukan saja ideal.

Dalam Optimasi Streaming Waktu Nyata, kami menambahkan bagian baru, [Mengoptimalkan Bitrate Audio dan Dukungan Stereo.](#)

[Stage Health: CloudWatch Metrik Baru](#)

CloudWatch Metrik yang dirilis untuk streaming real-time IVS. Lihat [Memantau Streaming Waktu Nyata IVS.](#)

12 Oktober 2023

Siaran SDK: Android 1.12.1	Nomor versi dan tautan artefak yang diperbarui untuk rilis baru, dalam panduan SDK real-time-streaming siaran: Android. Juga menambahkan bagian baru, Menggunakan Mikrofon Bluetooth .	12 Oktober 2023
	Halaman arahan dokumentasi Amazon IVS menunjuk ke versi Referensi SDK Siaran saat ini.	
	Lihat juga Catatan Rilis Amazon IVS untuk rilis ini.	
Siaran SDK: Web 1.5.2	Nomor versi dan tautan artefak yang diperbarui untuk rilis baru, dalam panduan SDK real-time-streaming siaran: Web.	14 September 2023
	Halaman arahan dokumentasi Amazon IVS menunjuk ke versi Referensi SDK Siaran saat ini.	
	Lihat juga Catatan Rilis Amazon IVS untuk rilis ini.	
Memulai dengan IVS Real-Time Streaming	Di Android > Instal Broadcast SDK , tambahkan data binding.	12 September 2023
Penanganan kesalahan SDK siaran	Menambahkan bagian “Penanganan Kesalahan” ke Panduan SDK Siaran: Web , Android , dan iOS .	12 September 2023

[Memulai dengan IVS Real-Time Streaming](#)

Di [Distribusikan Token Peserta](#), tambahkan catatan Penting tentang tidak membangun fungsionalitas berdasarkan format token saat ini.

1 September 2023

[Memulai dengan IVS Real-Time Streaming](#)

Dalam [Mengatur Izin IAM](#), perbarui kumpulan izin.

31 Agustus 2023

[Broadcast SDK: Web 1.5.1, Android 1.12.0, dan iOS 1.12.0](#)

[Nomor versi dan tautan artefak yang diperbarui untuk rilis baru, dalam panduan SDK real-time-streaming siaran: Web, Android, dan iOS.](#)

23 Agustus 2023

Pada [halaman arahan dokumentasi Amazon IVS](#), memperbarui tautan Referensi SDK siaran untuk menunjuk ke versi baru.

Lihat juga [Catatan Rilis](#) Amazon IVS untuk rilis ini.

[Peluncuran streaming waktu nyata](#)

Perubahan dokumentasi utama menyertai rilis ini. Kami mengganti nama dokumentasi sebelumnya menjadi IVS Low-Latency Streaming dan menerbitkan dokumentasi IVS Real-Time Streaming baru. [Halaman arahan dokumentasi IVS](#) sekarang memiliki bagian terpisah untuk streaming real-time dan streaming latensi rendah. Setiap bagian memiliki Panduan Pengguna dan Referensi API sendiri.

Agustus 7, 2023

Untuk perubahan dokumentasi lainnya, lihat [Riwayat Dokumen \(Streaming Latensi Rendah\)](#).

[Broadcast SDK: Web 1.5.0, Android 1.11.0, dan iOS 1.11.0](#)

[Nomor versi dan tautan artefak yang diperbarui untuk rilis baru, dalam panduan SDK siaran: Web, Android, dan iOS.](#)

Agustus 7, 2023

Pada [halaman arahan dokumentasi Amazon IVS](#), memperbarui tautan Referensi SDK siaran untuk menunjuk ke versi baru.

Lihat juga [Catatan Rilis Amazon IVS](#) untuk rilis ini.

Perubahan Referensi API Streaming Waktu Nyata IVS

Perubahan API	Deskripsi	Tanggal
Rekaman komposit	<p>Kami menambahkan 4 StorageConfiguration titik akhir dan 7 objek (DestinationDetail,, S3 Recording Configuration, S3DetailDestinationConfiguration, S3,,). StorageConfiguration StorageConfiguration StorageConfigurationSummary</p> <p>Kami memodifikasi 3 objek (Komposisi, Tujuan, DestinationConfiguration). Hal ini mempengaruhi GetComposition respon dan StartComposition permintaan dan respon.</p>	16 November 2023
Komposisi sisi server	Kami menambahkan 8 Komposisi dan EncoderConfiguration titik akhir dan 11 objek (ChannelDestinationConfiguration, Komposisi, CompositionSummary, Tujuan,, DestinationConfiguration, Destinati onSummary, EncoderConfiguration, EncoderCo nfigurationSummary, GridConfiguration LayoutCon figuration, dan Video).	16 November 2023
Stage Health: Data Peserta Baru	Menambahkan enam bidang ke objek Peserta :browserName ,browserVersion ,ispName,osName,osVersion , dansdkVersion . Ini mempengaruhi GetParticipant respons.	12 Oktober 2023
Token Peserta	Menambahkan catatan Penting tentang tidak membangun fungsionalitas berdasarkan format token saat ini.	1 September 2023
Peluncuran Streaming Waktu Nyata IVS	Perubahan dokumentasi utama menyertai rilis ini. Kami mengganti nama dokumentasi sebelumnya menjadi IVS Low-Latency Streaming dan menerbitkan dokumentasi IVS Real-Time Streaming baru. Halaman arahan dokumentasi IVS sekarang memiliki	Agustus 7, 2023

Perubahan API	Deskripsi	Tanggal
	<p>bagian terpisah untuk streaming real-time dan streaming latensi rendah. Setiap bagian memiliki Panduan Pengguna dan Referensi API sendiri.</p> <p>Referensi API Streaming Waktu Nyata IVS adalah bagian dari dokumentasi streaming real-time IVS. Sebelumnya itu berjudul IVS Stage API Reference . Sejarah sebelumnya dijelaskan dalam Sejarah Dokumen (Streaming Latensi Rendah).</p>	

Catatan Rilis (Streaming Waktu Nyata)

Februari 6, 2024

OBS dan WHIP Support

IVS dapat digunakan dengan encoder yang kompatibel dengan Whip seperti OBS untuk mempublikasikan ke streaming real-time IVS. WHIP (WebRTC-HTTP Ingestion Protocol) adalah draf IETF yang dikembangkan untuk membakukan konsumsi WebRTC. Lihat halaman baru di [OBS dan WHIP Support](#).

Februari 1, 2024

SDK Siaran Amazon IVS: Android 1.14.1, iOS 1.14.1, Web 1.8.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
Siaran Web SDK 1.8.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference</p> <ul style="list-style-type: none">• Pengkodean berlapis dengan simulcast sekarang dinonaktifkan secara default.• Memperbaiki masalah saat instance Stage tidak akan terputus dengan bersih saat Stage dihapus, atau saat peserta terputus dari server. SDK sekarang memancarkan <code>STAGE_CONNECTION_STATE_CHANGED</code> peristiwa dengan status <code>DISCONNECTED</code> (bukan <code>ERRORED</code> dan kemudian <code>CONNECTING</code>).• Memperbaiki masalah saat penerbitan akan gagal saat memperbarui strategi dengan trek audio atau video kosong.

Platform	Unduhan dan Perubahan
SDK Siaran Android 1.14.1	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs /1.14.1/android</p> <ul style="list-style-type: none">• Pengkodean berlapis dengan simulcast sekarang dinonaktifkan secara default.• Diperbarui <code>libWebRTC</code> dari M108 ke M119.• Memperbaiki beberapa crash untuk meningkatkan stabilitas keseluruhan.• Menambahkan dukungan untuk penerbitan stereo. Ini dapat diaktifkan melalui <code>StageAudioConfiguration</code> objek.• Memperbaiki bug yang menyebabkan umpan hitam dari peserta setelah bergabung dengan sesi.• <code>libWebRTC</code> Referensi internal yang diperbarui untuk menghindari konflik simbol ketika <code>libWebRTC</code> versi lain disertakan dalam aplikasi host yang sama.

Platform	Unduhan dan Perubahan
SDK Siaran iOS 1.14.1	<p>Unduh untuk streaming waktu nyata: https://broadcast.live-video.net/1.14.1/AmazonIVSBroadcast-Stages.xcframework.zip</p> <p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.14.1/ios</p> <ul style="list-style-type: none"> • Pengkodean berlapis dengan simulcast sekarang dinonaktifkan secara default. • Diperbarui <code>libWebRTC</code> dari M108 ke M119. • Memperbaiki beberapa crash untuk meningkatkan stabilitas keseluruhan. • Menambahkan dukungan untuk penerbitan stereo. Ini dapat diaktifkan melalui <code>IVSLocalStageStreamAudioConfiguration</code>. • Memperbaiki kerusakan saat mengaktifkan mode khusus audio untuk peserta lain. • Peningkatan TTV dan mengurangi ukuran biner.

Ukuran SDK Siaran: Android

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64-v8a	5.223 MB	13.118 MB
armeabi-v7a	4,524 MB	9.134 MB
x86_64	5,418 MB	13,955 MB
x86	5,61 MB	14.369 MB

Ukuran SDK Siaran: iOS

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64	3,350 MB	7,790 MB

Januari 3, 2024

SDK Siaran Amazon IVS: Android 1.13.4, iOS 1.13.4, Web 1.7.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Web 1.7.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference</p> <ul style="list-style-type: none"> Ditingkatkan time-to-video untuk pelanggan yang bergabung dengan tahapan. Menghapus minAudioBitrateKbps properti (itu tidak digunakan). Peningkatan pemulihan jaringan selama pemadaman atau perubahan internet.
SDK Siaran Android 1.13.4	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.13.4/android</p> <ul style="list-style-type: none"> StageAudioConfiguration sekarang mendukung pengaturan apakah pembatalan gema harus diaktifkan.
SDK Siaran iOS 1.13.4	<p>Unduh untuk streaming waktu nyata: https://broadcast.live-video.net/1.13.4/AmazonIVSBroadcast-Stages.xcframework.zip</p> <p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.13.4/ios</p>

Platform	Unduhan dan Perubahan
	<ul style="list-style-type: none"> • Di iOS, kami meningkatkan mesin audio untuk perekaman dan pemutaran dengan fokus pada stabilitas dan pemulihan. Ini meningkatkan dukungan untuk perubahan rute saat digunakan, meningkatkan pemulihan baterai untuk casing tepi, dan mengurangi jumlah pemblokiran utas utama. • Memperbaiki masalah di mana mikrofon mungkin tetap aktif bahkan setelah terlepas dari panggung, membiarkan indikator privasi iOS menyala. (SDK tidak memproses audio yang masuk pada saat itu.)

Ukuran SDK Siaran: Android

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64-v8a	5.187 MB	13.025 MB
armeabi-v7a	4,491 MB	9,056 MB
x86_64	5,359 MB	13.829 MB
x86	5,553 MB	14.214 MB

Ukuran SDK Siaran: iOS

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64	3,45 MB	7,84 MB

Desember 7, 2023

CloudWatch Metrik Baru

Kami mengganti nama metrik PacketLoss (Stage) menjadi DownloadPacketLoss (Stage). Kami juga merilis CloudWatch metrik tambahan untuk streaming real-time IVS:

- DownloadPacketLoss (Panggung, Peserta)
- DroppedFrames (Panggung, Peserta)
- SubscribeBitrate (Panggung, Peserta, MediaType)

Untuk detailnya, lihat [Memantau Streaming Waktu Nyata IVS](#).

Desember 4, 2023

SDK Siaran Amazon IVS: Android 1.13.2 dan iOS 1.13.2 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
Semua ponsel (Android dan iOS)	<ul style="list-style-type: none"> • Konfigurasi peredam bising tersedia bagi pengembang untuk mengaktifkan/menonaktifkan penerbitan.
SDK Siaran Android 1.13.2	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.13.2/android</p> <ul style="list-style-type: none"> • Meningkatkan waktu yang diperlukan untuk memuat video (TTV) saat bergabung dengan tahap pertama dalam sesi.
SDK Siaran iOS 1.13.2	<p>Unduh untuk streaming waktu nyata: https://broadcast.live-video.net/1.13.2/AmazonIVSBroadcast-Stages.xcframework.zip</p> <p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.13.2/ios</p>

Platform	Unduhan dan Perubahan
	<ul style="list-style-type: none"> Tidak ada perubahan dalam SDK real-time.

Ukuran SDK Siaran: Android

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64-v8a	5.177 MB	13,01 MB
armeabi-v7a	4,485 MB	9,045 MB
x86_64	5.352 MB	13.808 MB
x86	5,547 MB	14.192 MB

Ukuran SDK Siaran: iOS

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64	3,45 MB	7,82 MB

21 November 2023

SDK Siaran Amazon IVS: Android 1.13.1 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Android 1.13.1	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.13.1/android</p> <ul style="list-style-type: none"> Memperbaiki masalah yang menyebabkan an crash saat keluar, melepaskan, dan bergabung kembali dengan tahap yang sama dengan cepat.

Ukuran SDK Siaran: Android

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64-v8a	5.177 MB	13.102 MB
armeabi-v7a	4,485 MB	9,046 MB
x86_64	5.353 MB	13.809 MB
x86	5,547 MB	14.192 MB

17 November 2023

SDK Siaran Amazon IVS: Android 1.13.0 dan iOS 1.13.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
Semua ponsel (Android dan iOS)	<ul style="list-style-type: none"> • Optimasi Streaming yang Diperbarui. Antara lain, fitur “Adaptive Streaming: Layered Encoding with Simulcast” sekarang memerlukan opt-in eksplisit dan hanya didukung dalam versi SDK terbaru. • Meningkatkan stabilitas tahapan dengan mengurangi terjadinya crash langka. • Meningkatkan waktu yang diperlukan untuk memuat video (TTV) saat bergabung dengan panggung. • Meningkatkan pengalaman dengan perangkat Bluetooth. • Mengoptimalkan penggunaan CPU dan memori SDK, dan mengurangi ukuran perpustakaan.

Platform	Unduhan dan Perubahan
	<ul style="list-style-type: none">• Ditambahkan <code>StageAudioManager</code> kelas, yang dapat digunakan untuk mengatur pengambilan audio dan parameter pemutaran, termasuk preset untuk komunikasi suara, pemutaran media dan banyak lagi. Untuk detailnya, lihat halaman baru, IVS Broadcast SDK: Mode Audio Seluler.• Menambahkan <code>requestQualityStats</code> fungsi baru untuk menampilkan peristiwa kualitas terstruktur dari statistik WebRTC.• Menambahkan fungsi baru untuk memperbaiki bitrate audio. Ini diatur pada <code>LocalStageStream</code> objek seperti konfigurasi video, tetapi melalui objek konfigurasi audio baru.

Platform	Unduhan dan Perubahan
SDK Siaran Android 1.13.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs /1.13.0/android</p> <ul style="list-style-type: none">• Semua metode pada StageRenderer antarmuka sekarang opsional.• Menambahkan dukungan ke pratinjau Surfaceview berbasis untuk kinerja yang lebih baik. <code>getPreview</code> Metode yang ada dalam <code>Session</code> dan <code>StageStream</code> terus mengembalikan <code>subkelasTextureView</code> , tetapi ini dapat berubah dalam versi SDK masa depan.• Jika aplikasi Anda bergantung pada <code>TextureView</code> spesifik, Anda dapat melanjutkan tanpa perubahan. Anda juga dapat beralih dari <code>getPreview</code> <code>getPreviewTextureView</code> ke untuk mempersiapkan perubahan akhirnya dari apa yang <code>getPreview</code> dikembalikan default.• Jika aplikasi Anda tidak memerlukan <code>TextureView</code> secara khusus, kami sarankan beralih ke <code>getPreviewSurfaceView</code> untuk penggunaan CPU dan memori yang lebih rendah.• SDK sekarang mengimplementasikan jenis pratinjau baru <code>ImagePreviewSurfaceTarget</code> yang disebut yang berfungsi dengan objek Android Surface yang disediakan aplikasi. Ini bukan subkelas dari Android View, yang memberikan fleksibilitas yang lebih baik.• Memperbaiki kasus di mana <code>onFrame</code> callback untuk peserta jarak jauh dipanggil

Platform	Unduhan dan Perubahan
	<p>pada waktu yang salah dengan ukuran yang salah.</p> <ul style="list-style-type: none"> • <code>SurfaceSource # getInputSurface</code> sekarang dianotasi dengan <code>@Nullable</code>. Kode Anda harus memeriksa nya sebelum menggunakannya. • Ditambahkan <code>UserId</code> dan attribute <code>s keParticipantInfo .attribute s</code>. Properti <code>UserId</code> dan disematkan dalam token dan aplikasi dapat mengambilnya melalui <code>ParticipantInfo</code> setiap kali peserta bergabung. • Pengambilan kamera dan rendering pratinjau sekarang default ke 720 x 1280 atau resolusi publikasi (mana yang lebih besar) pada 15 fps. Anda dapat menyesuaikan resolusi dan/atau fps menggunakan <code>StageVideoConfiguration # setCameraCaptureQuality</code>. • <code>IllegalArgumentException</code> dilemparkan saat mengatur properti konfigurasi sekarang menyertakan nilai yang disediakan dalam pesan pengecualian.

Platform	Unduhan dan Perubahan
<p>SDK Siaran iOS 1.13.0</p>	<p>Unduh untuk streaming waktu nyata: https://broadcast.live-video.net/1.13.0/AmazonIVSBroadcast-Stages.xcframework.zip</p> <p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.13.0/ios</p> <ul style="list-style-type: none"> • Memperbaiki masalah saat SDK tidak mengubah konfigurasi video jika konfigurasi video diperbarui sebelum dipublikasikan. • Memasukkan perbaikan Google untuk kerentanan keamanan LibVPX (CVE-2023-5217). (Perhatikan bahwa Android SDK tidak memerlukan perubahan apa pun untuk masalah ini.) • Aplikasi yang menggunakan pustaka lain yang menyertakan tidak libWebRTC akan lagi memiliki konflik dengan IVS Broadcast SDK. • Semua metode pada <code>IVSStageRenderer</code> protokol sekarang ditandai <code>@optional</code>. • Mikrofon dan kamera yang dikembalikan oleh SDK kami sekarang memiliki urutan penyortiran yang terjamin, seperti yang didokumentasikan dalam SDK itu sendiri. • Beberapa kamera sekarang dapat memiliki nilai <code>true</code> untuk <code>isDefault</code> properti mereka, satu untuk setiap posisi yang ditentukan oleh sistem operasi. • Ditambahkan <code>IVSStageAudioManager</code>, yang memungkinkan kontrol yang tepat atas yang mendasarinya <code>AVAudioSession</code> untuk memungkinkan berbagai

Platform	Unduhan dan Perubahan
	<p>kasus penggunaan yang lebih luas untuk fungsionalitas Tahapan.</p> <ul style="list-style-type: none"> Menambahkan <code>UserId</code> ke <code>ParticipantInfo</code> .

Ukuran SDK Siaran: Android

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64-v8a	5,17 MB	13.00 MB
armeabi-v7a	4,48 MB	9,04 MB
x86_64	5,35 MB	13,80 MB
x86	5,54 MB	14.18 MB

Ukuran SDK Siaran: iOS

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64	3,45 MB	7,84 MB

16 November 2023

Rekaman Komposit

Fitur baru ini memungkinkan perekaman tampilan gabungan IVS Stage ke bucket Amazon S3. Lihat informasi yang lebih lengkap di:

- [Rekaman Komposit](#) - Ini adalah halaman baru.
- [Memulai Streaming Waktu Nyata IVS](#) - Kami menambahkan titik akhir S3 ke kebijakan di “Siapkan Izin IAM.”

- [Service Quotas - Kami menambahkan kuota](#) call-rate untuk endpoint baru.
- [Referensi API Streaming Waktu Nyata IVS](#) - Kami menambahkan 4 StorageConfiguration titik akhir dan 7 objek (DestinationDetail,, S3 RecordingConfiguration, S3DetailDestinationConfiguration, S3,,). StorageConfiguration StorageConfiguration StorageConfigurationSummary Kami juga memodifikasi 3 objek (Komposisi, Tujuan, DestinationConfiguration); ini memengaruhi GetComposition respons dan StartComposition permintaan dan respons.

16 November 2023

Komposisi Sisi Server

Komposisi sisi server IVS memungkinkan klien untuk menurunkan komposisi dan penyiaran tahap IVS ke layanan yang dikelola IVS. Komposisi sisi server dan siaran RTMP ke saluran dipanggil melalui titik akhir bidang kontrol IVS di wilayah asal panggung. Lihat informasi yang lebih lengkap di:

- [Memulai Streaming Waktu Nyata IVS](#) - Kami menambahkan titik akhir SSC ke kebijakan di “Siapkan Izin IAM.”
- [Menggunakan Amazon EventBridge dengan IVS Real-Time Streaming](#) — Kami menambahkan metrik baru.
- [Komposisi Sisi Server](#) - Dokumen baru ini mencakup ikhtisar dan instruksi penyiapan.
- [Service Quotas \(Real-Time Streaming\)](#) - Kami menambahkan batas call-rate baru dan kuota lainnya.
- [Referensi API Streaming Real-Time](#) - Kami menambahkan 8 Komposisi dan EncoderConfiguration titik akhir dan 11 objek (ChannelDestinationConfiguration, Komposisi CompositionSummary,, Tujuan DestinationConfiguration,, DestinationSummary, EncoderConfiguration, EncoderConfigurationSummary, GridConfiguration, LayoutConfiguration, dan Video).

Dalam Panduan Pengguna Streaming Latensi Rendah IVS, lihat:

- [Mengaktifkan Beberapa Host di Aliran IVS](#) - Kami menambahkan “Menyiarkan Panggung: Komposisi Sisi Klien versus Sisi Server” dan memperbarui “4. Siarkan Panggung.”

16 Oktober 2023

SDK Siaran Amazon IVS: Web 1.6.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Web 1.6.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference</p> <ul style="list-style-type: none"> • Peningkatan Time-To-Video (TTV). • Menambahkan <code>maxAudioBitrate</code> konfigurasi, mendukung hingga 128kbps saluran audio mono atau stereo.

12 Oktober 2023

CloudWatch Metrik Baru dan Data Peserta

Kami merilis CloudWatch metrik untuk streaming real-time IVS. Untuk detailnya, lihat [Memantau Streaming Waktu Nyata IVS](#).

Kami juga menambahkan enam bidang ke objek API

Peserta: `browserName`, `browserVersion`, `ispName`, `osName`, `osVersion`, dan `sdkVersion`. Ini mempengaruhi `GetParticipant` respons. Lihat [Referensi API Streaming Waktu Nyata IVS](#).

12 Oktober 2023

SDK Siaran Amazon IVS: Android 1.12.1 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Android 1.12.1	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.12.1/android</p>

Platform	Unduhan dan Perubahan
	<ul style="list-style-type: none"> Memperbaiki bug di mana panggilan <code>BroadcastSession.setListener</code> menghasilkan kesalahan.

Ukuran SDK Siaran: Android

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64-v8a	5,853 MB	16,375 MB
armeabi-v7a	4,895 MB	10.803 MB
x86_64	6.149 MB	17.318 MB
x86	6,328 MB	17.186 MB

14 September 2023

SDK Siaran Amazon IVS: Web 1.5.2 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Web 1.5.2	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference</p> <ul style="list-style-type: none"> Memperbaiki bug yang mencegah penerbitan ulang <code>refreshStrategy</code> saat status yang diterbitkan memasuki <code>ERRORED</code> status.

23 Agustus 2023

SDK Siaran Amazon IVS: Web 1.5.1, Android 1.12.0, dan iOS 1.12.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Web 1.5.1	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference</p> <ul style="list-style-type: none"> • Memperbaiki bug dengan tipe Maybe internal pada TypeScript 5. • Menambahkan deteksi yang lebih baik untuk dukungan Simulcast. • Memperbaiki dua kondisi balapan <code>refreshStrategy</code> saat mencoba mempublikasikan. • Memperbaiki kondisi balapan <code>refreshStrategy</code> saat mencoba memperbarui peserta untuk berlangganan.
Semua ponsel (Android dan iOS)	<ul style="list-style-type: none"> • Memperbaiki masalah langka di mana tindakan penerbitan tidak pernah selesai. • Meningkatkan stabilitas tahapan dengan mengurangi terjadinya crash langka. • Meningkatkan stabilitas tahapan dengan menyelesaikan masalah kondisi balapan yang disebabkan oleh gabung/cuti cepat. • Menambahkan <code>setOnFrameCallback</code> metode baru pada <code>ImageDevice</code>. Hal ini memungkinkan pengamatan saat bingkai melewati perangkat itu sendiri, memberikan wawasan tentang rasio aspek gambar terbaru. Metode ini juga dapat digunakan untuk mendeteksi kapan frame pertama

Platform	Unduhan dan Perubahan
	diberikan untuk peserta jarak jauh dalam suatu tahap.
SDK Siaran Android 1.12.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.12.0/android</p> <ul style="list-style-type: none"> • Android 9 sekarang didukung. • Peningkatan penggunaan dan kinerja CPU.
SDK Siaran iOS 1.12.0	<p>Unduh untuk streaming waktu nyata: https://broadcast.live-video.net/1.12.0/AmazonIVSBroadcast-Stages.xcframework.zip</p> <p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.12.0/ios</p> <ul style="list-style-type: none"> • Memperbaiki tanda tangan <code>IVSDeviceDiscovery.createAudioSourceWithName</code> untuk mengembalikan <code>IVSCustomAudioSource</code> bukan <code>IVSCustomImageSource</code>

Ukuran SDK Siaran: Android

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64-v8a	5,853 MB	16,375 MB
armeabi-v7a	4,895 MB	10.803 MB
x86_64	6.149 MB	17.318 MB
x86	6,328 MB	17.186 MB

Ukuran SDK Siaran: iOS

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64	5,06 MB	10,92 MB

Agustus 7, 2023

SDK Siaran Amazon IVS: Web 1.5.0, Android 1.11.0, dan iOS 1.11.0

Platform	Unduhan dan Perubahan
Siaran Web SDK 1.5.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference</p> <ul style="list-style-type: none"> Ditambahkan Simulcast - Ketika diaktifkan, fitur ini memungkinkan penerbit untuk mengirim lapisan video berkualitas tinggi dan rendah. Pelanggan secara otomatis memilih kualitas optimal mereka berdasarkan kondisi jaringan mereka. Lihat Mengoptimalkan Media.
Semua ponsel (Android dan iOS)	<p>Ditambahkan Simulcast - Ketika diaktifkan, fitur ini memungkinkan penerbit untuk mengirim lapisan video berkualitas tinggi dan rendah. Pelanggan secara otomatis memilih kualitas optimal mereka berdasarkan kondisi jaringan mereka. Lihat “Aktifkan/Nonaktifkan Pengkodean Berlapis dengan Simulcast” di Panduan SDK Siaran Android dan iOS.</p>
SDK Siaran Android 1.11.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.11.0/android</p> <ul style="list-style-type: none"> Memperbaiki masalah saat membuat banyak tahapan pada akhirnya menghasilkan crash.

Platform	Unduhan dan Perubahan
	(Jumlah tahapan yang tepat tergantung pada perangkat.)
SDK Siaran iOS 1.11.0	<p>Unduh untuk streaming waktu nyata: https://broadcast.live-video.net/1.11.0/AmazonIVSBroadcast-Stages.xcframework.zip</p> <p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.11.0/ios</p> <ul style="list-style-type: none"> • Memperbaiki tanda tangan <code>IVSDeviceDiscovery.createAudioSourceWithName</code> untuk kembali <code>IVSCustomAudioSource</code> alih-alih <code>IVSCustomImageSource</code>

Ukuran SDK Siaran: Android

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64-v8a	5,811 MB	16.186 MB
armeabi-v7a	4,857 MB	10,646 MB
x86_64	6,108 MB	17.122 MB
x86	6,289 MB	16,994 MB

Ukuran SDK Siaran: iOS

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64	5.030 MB	10.810 MB

Agustus 7, 2023

Streaming Waktu Nyata

Amazon Interactive Video Service (IVS) Real-Time Streaming memungkinkan Anda mengirimkan streaming langsung dengan latensi yang bisa di bawah 300 milidetik dari host ke pemirsa.

Perubahan dokumentasi utama menyertai rilis ini. [Halaman arahan dokumentasi IVS](#) sekarang memiliki bagian terpisah untuk streaming real-time dan streaming latensi rendah. Setiap bagian memiliki Panduan Pengguna dan Referensi API sendiri. Untuk detail dokumentasi, lihat Riwayat Dokumen (untuk perubahan dokumentasi [real-time](#) dan [latensi rendah](#)). Untuk streaming real-time, mulailah dengan [Panduan Pengguna Streaming Waktu Nyata IVS dan Referensi API Streaming Waktu Nyata IVS](#).

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.