



Panduan Pengguna Streaming Waktu Nyata

Amazon IVS



Amazon IVS: Panduan Pengguna Streaming Waktu Nyata

Copyright © 2026 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 pelanggan, atau dengan cara apa pun yang merendahkan atau mendiskreditkan Amazon. Semua merek dagang lain yang tidak dimiliki oleh Amazon merupakan hak milik masing-masing pemiliknya, yang mungkin atau mungkin tidak terafiliasi, terkait dengan, atau disponsori oleh Amazon.

Table of Contents

Apa itu Streaming Waktu Nyata IVS?	1
Solusi Global, Kontrol Regional	2
Streaming dan Penayangan bersifat 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
Langkah 1: 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
Langkah 2: Buat Panggung dengan Rekaman Peserta Opsional	10
Rekaman Peserta Individu	10
Instruksi Konsol	12
Instruksi CLI	16
Langkah 3: Bagikan Token Peserta	19
Membuat Token dengan Pasangan Kunci	19
Membuat Token dengan IVS Real-Time Streaming API	24
Langkah 4: Integrasikan SDK Siaran IVS	27
Web	27
Android	28
iOS	29
Langkah 5: Publikasikan dan Berlangganan Video	30
Konsol IVS	30
Web	31
Android	39
iOS	64
Memantau	95
Apa itu Sesi Panggung?	95
Lihat Sesi Panggung dan Peserta	95

Instruksi Konsol	95
Lihat Acara untuk Peserta	95
Instruksi Konsol	96
Instruksi CLI	96
CloudWatch Metrik Akses	97
CloudWatch Instruksi Konsol	97
Instruksi CLI	98
CloudWatch Metrik: Streaming Waktu Nyata IVS	98
SDK Siaran IVS	111
Persyaratan Platform	112
Platform Native	112
Peramban Desktop	112
Browser Seluler (iOS dan Android)	113
Tampilan Web	113
Akses Perangkat yang Diperlukan	113
Support	114
Penentuan versi	114
Panduan Web	115
Memulai	116
Menerbitkan dan Berlangganan	119
Masalah dan Solusi yang Diketahui	139
Penanganan Kesalahan	141
Panduan Android	144
Memulai	145
Menerbitkan dan Berlangganan	149
Masalah dan Solusi yang Diketahui	166
Penanganan Kesalahan	168
Panduan iOS	171
Memulai	171
Menerbitkan dan Berlangganan	173
Bagaimana iOS Memilih Resolusi Kamera dan Frame Rate	189
Masalah dan Solusi yang Diketahui	191
Penanganan Kesalahan	192
Perangkat Campuran	194
Terminologi	195
Perangkat Audio Campuran	197

Perangkat Gambar Campuran	197
Membuat dan Mengkonfigurasi Perangkat Gambar Campuran	200
Menghapus Sumber	202
Animasi dengan Transisi	203
Mencerminkan Siaran	204
Pertukaran Token	206
Bertukar Token	206
Menerima Pembaruan	207
Visibilitas Pembaruan	208
Sumber Gambar Kustom	209
Android	209
iOS	210
Sumber Audio Kustom	211
Android	211
Filter Kamera Pihak Ketiga	218
Mengintegrasikan Filter Kamera Pihak Ketiga	218
BytePlus	219
DeepAR	221
Jepret	221
Penggantian Latar Belakang	247
Mode Audio Seluler	268
Pengantar	269
Preset Mode Audio	270
Kasus Penggunaan Tingkat Lanjut	272
Integrasi dengan Lainnya SDKs	275
Menggunakan Amazon EventBridge dengan IVS	276
Membuat EventBridge Aturan Amazon untuk Amazon IVS	279
Contoh: Perubahan Status Komposisi	279
Contoh: Perubahan Status Pencatatan Peserta Individu	283
Contoh: Pembaruan Tahap	287
Komposisi Sisi Server	292
Gambaran umum	292
Manfaat	293
Siklus Hidup Komposisi	294
IVS API	295
Layout	296

Memulai	298
Prasyarat	299
Instruksi API	299
Instruksi CLI	300
Pemesanan Peserta Kustom	302
Cara Kerja Pemesanan Kustom	303
Membuat Token dengan Atribut Pemesanan	303
Contoh Kasus Penggunaan	304
Kompatibilitas Ke Belakang	304
Mengaktifkan Berbagi Layar	305
Buat Sumber EncoderConfiguration Daya	305
Mulai Komposisi	305
Hentikan Komposisi	307
Masalah dan Solusi yang Diketahui	308
Merekam	309
Rekaman Peserta Individu	309
Rekaman Komposit	310
Thumbnail	310
Rekaman Peserta Individu	311
Pengantar	311
Alur kerja	312
Rekaman Hanya Audio	315
Rekaman Khusus Thumbnail	316
Isi Rekaman	317
Gabungkan Rekaman Peserta Individu yang Terfragmentasi	318
Sinkronkan Beberapa Rekaman Peserta	319
File Metadata JSON	320
Mengonversi Rekaman ke MP4	327
Rekaman Komposit	327
Prasyarat	327
Contoh Perekaman Komposit: StartComposition dengan Tujuan Bucket S3	328
Isi Rekaman	330
Kebijakan Bucket untuk StorageConfiguration	331
File Metadata JSON	332
Pemutaran Konten Rekaman dari Bucket Pribadi	340
Pemecahan masalah	345

Masalah yang Diketahui	345
Aliran Ingest	346
Protokol yang Didukung	346
Spesifikasi Media yang Didukung	347
RTMP	348
Prasyarat	348
Video Trek Tunggal RTMP	349
E-RTMP Multitrack Video	350
Pencernaan Pribadi ke Tahapan	352
WHIP	352
Panduan OBS	353
Replikasi Peserta	354
Menggunakan Replikasi Peserta	355
Prasyarat	355
Mulai Replikasi Peserta	355
Hentikan Replikasi Peserta	356
Service Quotas	357
Peningkatan Kuota Layanan	357
Kuota Tingkat Panggilan API	357
Kuota Lainnya	359
Pengoptimalan Streaming	362
Pengantar	362
Streaming Adaptif: Pengkodean Berlapis dengan Simulcast	362
Lapisan Default, Kualitas, dan Framerate	363
Resolusi Lapisan	364
Mengkonfigurasi Layered Encoding dengan Simulcast (Publisher)	365
Mengkonfigurasi Layered Encoding dengan Simulcast (Subscriber)	366
Konfigurasi Streaming	366
Mengubah Bitrate Stream Video	366
Mengubah Framerate Stream Video	367
Mengoptimalkan Audio Bitrate dan Dukungan Stereo	368
Mengubah Buffer Jitter Pelanggan MinDelay	369
Optimasi yang Disarankan	371
Persyaratan Jaringan	372
Biasa	372
Media	372

Biaya	373
Sumber Daya & Support	374
Demo dan Sumber Daya Lainnya	374
Support	375
Glosarium	377
Riwayat Dokumen	400
Perubahan Panduan Pengguna Streaming Waktu Nyata	400
Perubahan Referensi API Streaming Waktu Nyata IVS	442
Catatan Rilis	449
Maret 12, 2026	449
SDK Siaran IVS: Web 1.33.0 (Streaming Waktu Nyata)	449
Maret 12, 2026	449
SDK Siaran Amazon IVS: Android 1.40.0, iOS 1.40.0 (Streaming Waktu Nyata)	449
Februari 13, 2026	451
SDK Siaran Amazon IVS: Android 1.39.0, iOS 1.39.0 (Streaming Waktu Nyata)	451
Februari 12, 2026	453
SDK Siaran IVS: Web 1.32.0 (Streaming Waktu Nyata)	453
Januari 13, 2026	453
SDK Siaran Amazon IVS: Android 1.38.0, iOS 1.38.0 (Streaming Waktu Nyata)	453
Desember 11, 2025	457
SDK Siaran Amazon IVS: Android 1.37.1 (Streaming Waktu Nyata)	457
Desember 9, 2025	458
Pertukaran Token Peserta	458
Desember 5, 2025	458
SDK Siaran IVS: Web 1.31.0 (Streaming Waktu Nyata)	458
Desember 5, 2025	459
SDK Siaran Amazon IVS: Android 1.37.0, iOS 1.37.0 (Streaming Waktu Nyata)	459
November 7, 2025	460
Sinkronisasi Rekaman Peserta Individu	460
Oktober 30, 2025	460
SDK Siaran IVS: Web 1.30.0 (Streaming Waktu Nyata)	460
Oktober 30, 2025	461
SDK Siaran Amazon IVS: Android 1.36.0, iOS 1.36.0 (Streaming Waktu Nyata)	461
Oktober 14, 2025	462
Batas Waktu Nyata yang Diperbarui: Komposisi	462
Oktober 2, 2025	462

SDK Siaran IVS: Web 1.29.0 (Streaming Waktu Nyata)	462
Oktober 2, 2025	463
SDK Siaran Amazon IVS: Android 1.35.0, iOS 1.35.0 (Streaming Waktu Nyata)	463
September 16, 2025	464
Komposisi Sisi Server Pemesanan Peserta Kustom	464
September 11, 2025	464
SDK Siaran Amazon IVS: Android 1.34.0, iOS 1.34.0 (Streaming Waktu Nyata)	464
September 10, 2025	466
Antarmuka VPC Endpoint	466
September 4, 2025	466
SDK Siaran IVS: Web 1.28.0 (Streaming Waktu Nyata)	466
Agustus 7, 2025	467
SDK Siaran IVS: Web 1.27.0 (Streaming Waktu Nyata)	467
Agustus 7, 2025	467
SDK Siaran Amazon IVS: Android 1.33.0, iOS 1.33.0 (Streaming Waktu Nyata)	467
Juli 25, 2025	469
SDK Siaran Amazon IVS: Android 1.32.2 (Streaming Waktu Nyata)	469
Juli 23, 2025	470
Penegakan Metrik dan Batasan Real-Time Baru: Penerbit dan Langganan Serentak	470
Juli 15, 2025	470
Batas Waktu Nyata Baru: Replikasi Peserta Serentak	470
Juli 10, 2025	470
SDK Siaran Amazon IVS: Android 1.32.1, iOS 1.32.1 (Streaming Waktu Nyata)	470
Juli 7, 2025	472
SDK Siaran IVS: Web 1.26.0 (Streaming Waktu Nyata)	472
Juni 23, 2025	473
Metrik dan Batas Waktu Nyata Baru: Penerbit dan Langganan Bersamaan	473
Juni 20, 2025	473
E-RTMP Multitrack Video Insert Support	473
Juni 16, 2025	474
SDK Siaran IVS: Web 1.25.1 (Streaming Waktu Nyata)	474
Juni 12, 2025	474
SDK Siaran Amazon IVS: Android 1.31.0, iOS 1.31.0 (Streaming Waktu Nyata)	474
Juni 12, 2025	475
SDK Siaran IVS: Web 1.25.0 (Streaming Waktu Nyata)	475
29 Mei 2025	476

Replikasi Peserta	476
26 Mei 2025	476
SDK Siaran Amazon IVS: Android 1.30.1 (Streaming Waktu Nyata)	476
15 Mei 2025	477
SDK Siaran IVS: Web 1.24.0 (Streaming Waktu Nyata)	477
15 Mei 2025	477
SDK Siaran Amazon IVS: Android 1.30.0, iOS 1.30.0 (Streaming Waktu Nyata)	477
2 Mei 2025	478
SDK Siaran IVS: Web 1.23.1 (Streaming Waktu Nyata)	478
17 April 2025	479
SDK Siaran Amazon IVS: Android 1.29.0, iOS 1.29.0 (Streaming Waktu Nyata)	479
17 April 2025	480
SDK Siaran IVS: Web 1.23.0 (Streaming Waktu Nyata)	480
April 2, 2025	481
Kuota Baru: Komposisi Per Tahap	481
Maret 20, 2025	481
SDK Siaran Amazon IVS: Android 1.28.1, iOS 1.28.1 (Streaming Waktu Nyata)	481
Maret 20, 2025	482
SDK Siaran IVS: Web 1.22.0 (Streaming Waktu Nyata)	482
Maret 19, 2025	483
SDK Siaran Amazon IVS: Android 1.27.2, iOS 1.27.2 (Streaming Waktu Nyata)	483
Maret 13, 2025	484
Durasi Segmen Target	484
Maret 6, 2025	485
Jahitan Rekaman Peserta Individu	485
Maret 3, 2025	485
SDK Siaran Amazon IVS: iOS 1.27.1 (Streaming Waktu Nyata)	485
Februari 20, 2025	486
SDK Siaran Amazon IVS: Android 1.27.0, iOS 1.27.0 (Streaming Waktu Nyata)	486
Februari 20, 2025	487
SDK Siaran IVS: Web 1.21.0 (Streaming Waktu Nyata)	487
Januari 30, 2025	488
SDK Siaran Amazon IVS: Android 1.26.0, iOS 1.26.0 (Streaming Waktu Nyata)	488
Januari 23, 2025	489
SDK Siaran IVS: Web 1.20.0 (Streaming Waktu Nyata)	489
Desember 12, 2024	489

SDK Siaran Amazon IVS: Android 1.25.0, iOS 1.25.0 (Streaming Waktu Nyata)	489
Desember 12, 2024	492
SDK Siaran IVS: Web 1.19.0 (Streaming Waktu Nyata)	492
Desember 10, 2024	492
Konfigurasi Thumbnail Streaming Waktu Nyata	492
November 13, 2024	493
SDK Siaran Amazon IVS: Android 1.24.0, iOS 1.24.0 (Streaming Waktu Nyata)	493
November 12, 2024	494
SDK Siaran IVS: Web 1.18.0 (Streaming Waktu Nyata)	494
Oktober 10, 2024	495
SDK Siaran IVS: Web 1.17.0 (Streaming Waktu Nyata)	495
Oktober 10, 2024	495
SDK Siaran Amazon IVS: Android 1.23.0, iOS 1.23.0 (Streaming Waktu Nyata)	495
September 11, 2024	496
SDK Siaran Amazon IVS: Android 1.22.0, iOS 1.22.0 (Streaming Waktu Nyata)	496
September 11, 2024	497
SDK Siaran IVS: Web 1.16.0 (Streaming Waktu Nyata)	497
September 9, 2024	498
Tertelan RTMP	498
Agustus 19, 2024	498
Terbitkan/Berlangganan Dalam Konsol	498
Agustus 15, 2024	498
SDK Siaran IVS: Web 1.15.0 (Streaming Waktu Nyata)	498
Agustus 15, 2024	499
SDK Siaran Amazon IVS: Android 1.21.0, iOS 1.21.0 (Streaming Waktu Nyata)	499
Juli 18, 2024	501
SDK Siaran IVS: Web 1.14.0 (Streaming Waktu Nyata)	501
Juli 18, 2024	501
SDK Siaran Amazon IVS: Android 1.20.0, iOS 1.20.0 (Streaming Waktu Nyata)	501
Juni 26, 2024	502
Hasilkan Token Peserta dengan Pasangan Kunci	502
Juni 20, 2024	503
Rekaman Peserta Individu	503
Juni 13, 2024	503
SDK Siaran Amazon IVS: Android 1.19.0, iOS 1.19.0 (Streaming Waktu Nyata)	503
Juni 13, 2024	504

SDK Siaran IVS: Web 1.13.0 (Streaming Waktu Nyata)	504
Mei 20, 2024	505
SDK Siaran IVS: Web 1.12.0 (Streaming Waktu Nyata)	505
16 Mei 2024	506
SDK Siaran Amazon IVS: Android 1.18.0, iOS 1.18.0 (Streaming Waktu Nyata)	506
6 Mei 2024	507
SDK Siaran IVS: Web 1.11.0 (Streaming Waktu Nyata)	507
April 30, 2024	508
SDK Siaran IVS: Web 1.10.1 (Streaming Waktu Nyata)	508
April 30, 2024	508
SDK Siaran Amazon IVS: Android 1.15.2, iOS 1.15.2 (Streaming Waktu Nyata)	508
April 22, 2024	509
SDK Siaran Amazon IVS: Android 1.17.0, iOS 1.17.0 (Streaming Waktu Nyata)	509
Maret 21, 2024	511
SDK Siaran Amazon IVS: Android 1.16.0, iOS 1.16.0, Web 1.10.0 (Streaming Waktu Nyata)	511
Maret 13, 2024	512
SDK Siaran Amazon IVS: Android 1.15.1, iOS 1.15.1 (Streaming Waktu Nyata)	512
Maret 13, 2024	513
Pembaruan API Komposisi Sisi Server	513
8 Maret 2024	514
Pembaruan Tata Letak Komposisi Sisi Server	514
Februari 22, 2024	514
SDK Siaran Amazon IVS: Android 1.15.0, iOS 1.15.0, Web 1.9.0 (Streaming Waktu Nyata)	514
Februari 7, 2024	515
Pembaruan Tata Letak Komposisi Sisi Server	515
Februari 6, 2024	518
OBS dan WHIP Support	518
Februari 1, 2024	518
SDK Siaran Amazon IVS: Android 1.14.1, iOS 1.14.1, Web 1.8.0 (Streaming Waktu Nyata)	518
Januari 3, 2024	521
SDK Siaran Amazon IVS: Android 1.13.4, iOS 1.13.4, Web 1.7.0 (Streaming Waktu Nyata)	521
Desember 7, 2023	523

CloudWatch Metrik Baru	523
Desember 4, 2023	523
SDK Siaran Amazon IVS: Android 1.13.2 dan iOS 1.13.2 (Streaming Waktu Nyata)	523
21 November 2023	524
SDK Siaran Amazon IVS: Android 1.13.1 (Streaming Waktu Nyata)	524
17 November 2023	525
SDK Siaran Amazon IVS: Android 1.13.0 dan iOS 1.13.0 (Streaming Waktu Nyata)	525
16 November 2023	530
Rekaman Komposit	530
16 November 2023	531
Komposisi Sisi Server	531
16 Oktober 2023	532
SDK Siaran Amazon IVS: Web 1.6.0 (Streaming Waktu Nyata)	532
12 Oktober 2023	532
CloudWatch Metrik Baru dan Data Peserta	532
12 Oktober 2023	532
SDK Siaran Amazon IVS: Android 1.12.1 (Streaming Waktu Nyata)	532
14 September 2023	533
SDK Siaran Amazon IVS: Web 1.5.2 (Streaming Waktu Nyata)	533
23 Agustus 2023	534
SDK Siaran Amazon IVS: Web 1.5.1, Android 1.12.0, dan iOS 1.12.0 (Streaming Waktu Nyata)	534
Agustus 7, 2023	536
SDK Siaran Amazon IVS: Web 1.5.0, Android 1.11.0, dan iOS 1.11.0	536
Agustus 7, 2023	538
Streaming Waktu Nyata	538
.....	dxxxix

Apa itu Streaming Waktu Nyata Amazon IVS?

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

Kekuatan:

- Latensi real-time — Buat aplikasi untuk kasus penggunaan yang sensitif terhadap latensi, membantu pemirsa Anda tetap terhubung dan terlibat dengan streaming real-time IVS. Kirimkan streaming langsung dengan latensi yang bisa di bawah 300 milidetik dari host ke pemirsa.
- Konkurensi tinggi - Buka potensi interaksi skala besar dengan streaming real-time IVS. Akomodasi pemirsa melebihi 25.000 pemirsa dan memungkinkan hingga 12 host untuk naik panggung virtual. (Untuk batasan default dan instruksi tentang meminta peningkatan, lihat Service Quotas [untuk streaming real-time dan streaming latensi rendah](#).)
- Dioptimalkan untuk seluler - Streaming real-time IVS dioptimalkan untuk kasus penggunaan seluler, melayani beragam perangkat dan kemampuan jaringan. Dengan mengintegrasikan siaran Amazon IVS SDKs untuk Android dan iOS, pengguna Anda dapat terlibat 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 waktu nyata.
- Mode Versus (VS) — Hasilkan pengalaman dengan side-by-side kompetisi dan biarkan pemirsa menonton tuan rumah bersaing secara real-time.
- Ruang audio — Undang pendengar untuk bergabung dalam percakapan sebagai tamu dan kembangkan 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 diterbitkan (demo, contoh kode, posting blog), memperkirakan biaya, dan mengalami Amazon IVS melalui demo langsung.

Solusi Global, Kontrol Regional

Streaming dan Penayangan bersifat Global

Anda dapat menggunakan Amazon IVS untuk melakukan 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. Bidang data mengacu streaming/ingesting dan melihat.

Kontrol adalah Regional

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

Cara lain untuk mengatakan ini adalah bahwa Amazon IVS adalah “AWS layanan regional.” Artinya, sumber daya Amazon IVS di setiap wilayah tidak tergantung 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 dibuat.

Jika Anda menggunakan an...	Anda menentukan wilayah dengan...
Konsol Amazon IVS	Menggunakan drop-down Pilih Wilayah di kanan atas bilah navigasi.
Amazon IVS API	Menggunakan titik akhir layanan yang sesuai. Lihat Referensi API Streaming Waktu Nyata Amazon IVS . (Jika Anda mengakses API melalui SDK, atur parameter SDK. <code>region</code> Lihat Alat untuk Dibangun AWS .)

Jika Anda menggunakan...	Anda menentukan wilayah dengan...
AWS CLI	Baik: <ul style="list-style-type: none">• Menambahkan <code>--region <aws-region></code> ke perintah CLI Anda.• Menempatkan wilayah di file AWS konfigurasi lokal Anda.

Ingat, terlepas dari wilayah tempat 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 Streaming Waktu Nyata IVS](#)
- [Langkah 1: Mengatur Izin IAM](#)
- [Langkah 2: Buat Panggung dengan Rekaman Peserta Opsional](#)
- [Langkah 3: Bagikan Token Peserta](#)
- [Langkah 4: Integrasikan SDK Siaran IVS](#)
- [Langkah 5: Publikasikan dan Berlangganan Video](#)

Pengantar Streaming Waktu Nyata IVS

Bagian ini mencantumkan prasyarat untuk menggunakan streaming waktu nyata dan memperkenalkan terminologi utama.

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	
Stage	Ruang virtual tempat peserta dapat bertukar video secara real time.	
Host	Seorang peserta yang mengirim video lokal ke panggung.	
Pemirsa	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. [Siapkan 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: [konsol IVS](#), [the section called “Web”](#), [the section called “Android”](#), dan [the section called “iOS”](#).

Langkah 1: 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 yang dikelola 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 operasi IVS Get dan List API (untuk streaming latensi rendah dan real-time).
- Gunakan kebijakan terkelola `IVSFullAccess` AWS untuk memberi pengembang aplikasi Anda akses ke semua operasi API IVS (untuk streaming latensi rendah dan real-time).

Opsional: Buat Kebijakan Kustom untuk Izin Amazon IVS

Ikuti langkah-langkah ini:

1. Masuk ke AWS Management Console 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 add/delete (Izinkan/Tolak) izin akses operasi sesuai kebutuhan. Lihat [Referensi API Streaming Waktu Nyata IVS](#) untuk detail tentang operasi IVS.)

JSON

```
{
  "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 AWS Management Console 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, konfirmasi bahwa kebijakan IVS telah ditambahkan.

Langkah 2: Buat Panggung dengan Rekaman Peserta Opsional

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` operasi.

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.

Anda dapat membuat panggung - dengan atau tanpa rekaman peserta individu - melalui konsol Amazon IVS atau AWS CLI. Pembuatan dan perekaman panggung dibahas di bawah ini.

Rekaman Peserta Individu

Anda memiliki opsi untuk memungkinkan perekaman peserta individu untuk sebuah panggung. Jika perekaman peserta individu ke fitur S3 diaktifkan, semua siaran peserta individu ke panggung direkam dan disimpan ke bucket penyimpanan Amazon S3 yang Anda miliki. Selanjutnya, rekaman tersedia untuk pemutaran sesuai permintaan.

Menyiapkan ini adalah opsi lanjutan. Secara default, perekaman dinonaktifkan saat panggung dibuat.

Sebelum Anda dapat mengatur panggung untuk merekam, Anda harus membuat konfigurasi penyimpanan. Ini adalah sumber daya yang menentukan lokasi Amazon S3 tempat aliran yang direkam untuk panggung disimpan. Anda dapat membuat dan mengelola konfigurasi penyimpanan menggunakan konsol atau CLI; kedua prosedur diberikan di bawah ini. Setelah Anda membuat konfigurasi penyimpanan, Anda mengaitkannya dengan tahap baik saat Anda membuat panggung (seperti yang dijelaskan di bawah) atau yang lebih baru, dengan memperbarui tahap yang ada. (Di API, lihat [CreateStage](#) dan [UpdateStage](#).) Anda dapat mengaitkan beberapa tahap dengan konfigurasi penyimpanan yang sama. Anda dapat menghapus konfigurasi penyimpanan yang tidak lagi terkait dengan tahapan apa pun.

Ingatlah kendala berikut:

- Anda harus memiliki ember S3. Artinya, akun yang menyiapkan panggung untuk direkam harus memiliki bucket S3 tempat rekaman akan disimpan.
- Panggung, konfigurasi penyimpanan, dan lokasi S3 harus berada di AWS wilayah yang sama. Jika Anda membuat tahapan di wilayah lain dan ingin merekamnya, Anda juga harus menyiapkan konfigurasi penyimpanan dan bucket S3 di wilayah tersebut.

Merekam ke bucket S3 Anda memerlukan otorisasi dengan kredensi AWS Anda. Untuk memberikan IVS akses yang diperlukan, AWS IAM [Service-Linked Role](#) (SLR) dibuat secara otomatis saat konfigurasi perekaman dibuat: SLR dibatasi untuk memberikan izin tulis IVS hanya pada bucket tertentu.

Perhatikan bahwa masalah jaringan antara lokasi streaming dan AWS atau di dalam AWS dapat mengakibatkan beberapa kehilangan data saat merekam streaming Anda. Dalam kasus ini, Amazon IVS memprioritaskan streaming langsung daripada rekaman. Untuk redundansi, rekam secara lokal melalui alat streaming Anda.

Untuk informasi selengkapnya (termasuk cara mengatur pasca-pemrosesan atau pemutaran VOD pada file yang direkam), lihat Perekaman [Peserta Individu](#).

Cara Menonaktifkan Perekaman

Untuk menonaktifkan perekaman Amazon S3 pada tahap yang ada:

- Konsol — Pada halaman detail untuk tahap yang relevan, di bagian Rekam streaming peserta individu, matikan Aktifkan perekaman otomatis di bawah Rekam otomatis ke S3, lalu pilih Simpan perubahan. Ini menghapus asosiasi konfigurasi penyimpanan dengan panggung; aliran pada tahap itu tidak akan lagi direkam.
- CLI — Jalankan `update-stage` perintah dan teruskan ARN konfigurasi perekaman sebagai string kosong:

```
aws ivs-realtime update-stage --arn arn:aws:ivs:us-west-2:123456789012:stage/abcdABCDefgh --auto-participant-recording-configuration storageConfigurationArn=""
```

Ini mengembalikan objek panggung dengan string kosong untuk `storageConfigurationArn`, menunjukkan bahwa rekaman dinonaktifkan.

Petunjuk Konsol untuk Membuat Panggung IVS

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

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

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

☰ [Amazon IVS](#) > [Real-time](#) > [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 Real-Time works

Setup

Stage name – *optional*

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

Record individual participants [Info](#)

Auto-record to S3

Individual recordings will automatically be created in the attached S3 bucket for each publisher.

Enable automatic recording

▶ 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.
4. Jika Anda ingin mengaktifkan perekaman peserta individu, selesaikan langkah-langkah dalam [Mengatur Rekaman Peserta Individu Otomatis ke Amazon S3 \(Opsional\)](#) di bawah ini.
5. Pilih Buat tahap untuk membuat panggung. Halaman detail panggung muncul, untuk tahap baru.

Mengatur Rekaman Peserta Individu Otomatis ke Amazon S3 (Opsional)

Ikuti langkah-langkah ini untuk mengaktifkan perekaman peserta individu saat membuat panggung:

1. Pada halaman Buat panggung, di bawah Rekam peserta individu, aktifkan Aktifkan perekaman otomatis. Tampilan bidang tambahan, untuk memilih jenis media yang direkam, untuk memilih konfigurasi Penyimpanan yang ada atau membuat yang baru, dan untuk memilih apakah akan merekam thumbnail pada interval tertentu.

Record individual participants [Info](#)

Auto-record to S3
Individual recordings will automatically be created in the attached S3 bucket for each publisher.

Enable automatic recording

Record participant replicas
When enabled, replica participants will be recorded if they are replicated to this stage. This may result in duplicate recordings if auto-record to S3 is also enabled on the replica participant's source stage.

Enable replica recording

Recorded media types
Select the media types that will be recorded. By default, both audio and video will be recorded.

Audio and video

Storage configuration
Define the Amazon S3 bucket for the output

Choose an existing storage configuration

Target segment duration
Determines the target duration for recorded segments. Default: 6

6

Must be an integer greater than 1 and up to 10.

Merge fragmented recordings
In the event of a participant disconnect, Amazon IVS will merge the fragmented recordings into a single recording.

Reconnect in a window

Thumbnail recording

Record at an interval

Associated costs
There are four cost components to consider when enabling record to S3: storage, request and data retrieval, data transfer, and data management.

If you use a third-party encoder to publish content to this stage, **set your keyframe interval to 2 seconds to prevent playback issues.** It is not necessary to set the keyframe interval when using the IVS Broadcast SDKs.

2. Pilih jenis media mana yang akan direkam.
3. Pilih Buat konfigurasi penyimpanan. Jendela baru terbuka, dengan opsi untuk membuat ember Amazon S3 dan melampirkannya ke konfigurasi perekaman baru.

Create storage configuration [Info](#)

Setup

Storage configuration name – optional


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

Storage

- Create a new Amazon S3 bucket
 Select an existing Amazon S3 bucket

Bucket name

The bucket name must be unique and must not contain spaces or uppercase letters. [See rules for bucket naming](#).

 This bucket will be created with default permissions in the current region: **US East (N. Virginia) us-east-1**
[Choosing a region](#). [Permissions in Amazon S3](#)

▶ 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 storage configuration](#)

4. Isi kolom:

- a. Secara opsional masukkan nama konfigurasi Penyimpanan.
 - b. Masukkan nama Bucket.
5. Pilih Buat konfigurasi penyimpanan, untuk membuat sumber daya konfigurasi penyimpanan baru dengan ARN unik. Biasanya, pembuatan konfigurasi perekaman membutuhkan beberapa detik, tetapi bisa sampai 20 detik. Ketika konfigurasi penyimpanan dibuat, Anda dikembalikan ke jendela Create stage. Di sana, area Rekam peserta individu menunjukkan konfigurasi Penyimpanan baru Anda dan bucket S3 (Penyimpanan) yang Anda buat.

Record individual participants [Info](#)

Auto-record to S3

Individual recordings will automatically be created in the attached S3 bucket for each publisher.

Enable automatic recording

Record participant replicas

When enabled, replica participants will be recorded if they are replicated to this stage. This may result in duplicate recordings if auto-record to S3 is also enabled on the replica participant's source stage.

Enable replica recording

Recorded media types

Select the media types that will be recorded. By default, both audio and video will be recorded.

Audio and video

Storage configuration

Define the Amazon S3 bucket for the output

storage-configuration-1



[Create storage configuration](#)

Storage configuration name

storage-configuration-1

Storage

[s3://recording-configuration-bucket/](#)

Target segment duration

Determines the target duration for recorded segments. Default: 6

6

Must be an integer greater than 1 and up to 10.

Merge fragmented recordings

In the event of a participant disconnect, Amazon IVS will merge the fragmented recordings into a single recording.

Reconnect in a window

Thumbnail recording

Record at an interval

[Associated costs](#)

There are four cost components to consider when enabling record to S3: storage, request and data retrieval, data transfer, and data management.

[If you use a third-party encoder to publish content to this stage, set your keyframe interval to 2 seconds to prevent playback issues. It is not necessary to set the keyframe interval when using the IVS Broadcast SDKs.](#)

6. Anda dapat secara opsional mengaktifkan opsi non-default lainnya seperti merekam replika peserta, menggabungkan rekaman peserta individu, dan perekaman thumbnail.

Record individual participants [Info](#)

Auto-record to S3
Individual recordings will automatically be created in the attached S3 bucket for each publisher.

Enable automatic recording

Record participant replicas
When enabled, replica participants will be recorded if they are replicated to this stage. This may result in duplicate recordings if auto-record to S3 is also enabled on the replica participant's source stage.

Enable replica recording

Recorded media types
Select the media types that will be recorded. By default, both audio and video will be recorded.

Audio and video

Storage configuration
Define the Amazon S3 bucket for the output

storage-configuration-1 [Create storage configuration](#)

Storage configuration name
storage-configuration-1

Storage
[s3://recording-configuration-bucket/](#)

Target segment duration
Determines the target duration for recorded segments. Default: 6

6
Must be an integer greater than 1 and up to 10.

Merge fragmented recordings
In the event of a participant disconnect, Amazon IVS will merge the fragmented recordings into a single recording.

Reconnect in a window

Reconnect window (seconds)
Maximum gap in seconds between stream stops and restarts. [Learn more](#)

30
Must be an integer greater than 0 and up to 300.

Thumbnail recording

Record at an interval

Target thumbnail interval (seconds)
Determines how often thumbnails will be saved. Default: 60

60
Must be an integer greater than 0 and up to 86400.

Thumbnail storage

Store thumbnails sequentially
Record thumbnails in-order as unique files.

Associated costs
There are four cost components to consider when enabling record to S3: storage, request and data retrieval, data transfer, and data management.

If you use a third-party encoder to publish content to this stage, set your keyframe interval to 2 seconds to prevent playback issues. It is not necessary to set the keyframe interval when using the IVS Broadcast SDKs.

Petunjuk CLI untuk Membuat Tahap IVS

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

Sekarang Anda dapat menggunakan CLI untuk membuat dan mengelola sumber daya mengikuti salah satu dari dua prosedur di bawah ini, tergantung pada apakah Anda ingin membuat panggung dengan atau tanpa rekaman peserta individu diaktifkan.

Buat Panggung tanpa Rekaman Peserta Individu

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"
  }
}
```

Buat Panggung dengan Rekaman Peserta Individu

Untuk membuat panggung dengan rekaman peserta individu diaktifkan:

```
aws ivs-realtime create-stage --name "test-stage-participant-recording" --auto-
participant-recording-configuration storageConfigurationArn=arn:aws:ivs:us-
west-2:123456789012:storage-configuration/LKZ6QR7r55c2,mediaTypes=AUDIO_VIDEO
```

Secara opsional, teruskan `thumbnailConfiguration` parameter untuk mengatur mode penyimpanan dan perekaman thumbnail secara manual, dan detik interval thumbnail:

```
aws ivs-realtime create-stage --name "test-stage-participant-recording" --auto-
participant-recording-configuration storageConfigurationArn=arn:aws:ivs:us-
west-2:123456789012:storage-configuration/
LKZ6QR7r55c2,mediaTypes=AUDIO_VIDEO,thumbnailConfiguration="{targetIntervalSeconds=10,storage=
```

Secara opsional, teruskan `recordingReconnectWindowSeconds` parameter untuk mengaktifkan menggabungkan rekaman peserta individu yang terfragmentasi:

```
aws ivs-realtime create-stage --name "test-stage-participant-recording" --auto-
participant-recording-configuration "storageConfigurationArn=arn:aws:ivs:us-
```

```
west-2:123456789012:storage-configuration/  
LKZ6QR7r55c2,mediaTypes=AUDIO_VIDEO,thumbnailConfiguration="{targetIntervalSeconds=10,storage=[
```

Tanggapannya adalah:

```
{  
  "stage": {  
    "arn": "arn:aws:ivs:us-west-2:123456789012:stage/VSWjvX5X0kU3",  
    "autoParticipantRecordingConfiguration": {  
      "hlsConfiguration": {  
        "targetSegmentDurationSeconds": 6  
      },  
      "mediaTypes": [  
        "AUDIO_VIDEO"  
      ],  
      "recordingReconnectWindowSeconds": 60,  
      "recordParticipantReplicas": true,  
      "storageConfigurationArn": "arn:aws:ivs:us-west-2:123456789012:storage-  
configuration/LKZ6QR7r55c2",  
      "thumbnailConfiguration": {  
        "recordingMode": "INTERVAL",  
        "storage": [  
          "SEQUENTIAL",  
          "LATEST"  
        ],  
        "targetIntervalSeconds": 10  
      }  
    },  
    "endpoints": {  
      "events": "<events-endpoint>",  
      "rtmp": "<rtmp-endpoint>",  
      "rtmps": "<rtmps-endpoint>",  
      "whip": "<whip-endpoint>"  
    },  
    "name": "test-stage-participant-recording"  
  }  
}
```

Langkah 3: Bagikan Token Peserta

Sekarang setelah Anda memiliki panggung, Anda perlu membuat token dan mendistribusikannya kepada peserta, untuk memungkinkan peserta bergabung dengan panggung dan mulai mengirim dan menerima video. Ada dua pendekatan untuk menghasilkan token:

- [Buat](#) token dengan key pair.
- [Buat token dengan real-time-streaming API IVS](#).

Kedua pendekatan ini dijelaskan di bawah ini.

Membuat Token dengan Pasangan Kunci

Anda dapat membuat token pada aplikasi server Anda dan mendistribusikannya kepada peserta untuk bergabung dengan panggung. Anda perlu membuat public/private key pair ECDSA untuk menandatangani JWTs dan mengimpor kunci publik ke IVS. Kemudian IVS dapat memverifikasi token pada saat tahap bergabung.

IVS tidak menawarkan kedaluwarsa kunci. Jika kunci pribadi Anda dikompromikan, Anda harus menghapus kunci publik lama.

Buat Pasangan Kunci Baru

Ada berbagai cara untuk membuat key pair. Di bawah ini, kami memberikan dua contoh.

Untuk membuat key pair baru di konsol, ikuti langkah-langkah berikut:

1. Buka [konsol Amazon IVS](#). Pilih wilayah panggung Anda jika Anda belum berada di sana.
2. Di menu navigasi kiri, pilih Streaming waktu nyata > Kunci publik.
3. Pilih Buat kunci publik. Dialog Create public key muncul.
4. Ikuti petunjuknya dan pilih Buat.
5. Amazon IVS menghasilkan key pair baru. Kunci publik diimpor sebagai sumber daya kunci publik dan kunci pribadi segera tersedia untuk diunduh. Kunci publik juga dapat diunduh nanti jika perlu.

Amazon IVS menghasilkan kunci di sisi klien dan tidak menyimpan kunci pribadi. Pastikan Anda menyimpan kuncinya; Anda tidak dapat mengambilnya nanti.

[Untuk membuat key pair P384 EC baru dengan OpenSSL \(Anda mungkin harus menginstal OpenSSL terlebih dahulu\), ikuti langkah-langkah ini.](#) Proses ini memungkinkan Anda untuk mengakses kunci pribadi dan publik. Anda memerlukan kunci publik hanya jika Anda ingin menguji verifikasi token Anda.

```
openssl ecparam -name secp384r1 -genkey -noout -out priv.pem
openssl ec -in priv.pem -pubout -out public.pem
```

Sekarang impor kunci publik baru Anda, menggunakan petunjuk di bawah ini.

Impor Kunci Publik

Setelah Anda memiliki key pair, Anda dapat mengimpor kunci publik ke IVS. Kunci pribadi tidak diperlukan oleh sistem kami tetapi digunakan oleh Anda untuk menandatangani token.

Untuk mengimpor kunci publik yang ada dengan konsol:

1. Buka [konsol Amazon IVS](#). Pilih wilayah panggung Anda jika Anda belum berada di sana.
2. Di menu navigasi kiri, pilih Streaming waktu nyata > Kunci publik.
3. Pilih Impor. Dialog kunci publik Impor muncul.
4. Ikuti petunjuknya dan pilih Impor.
5. Amazon IVS mengimpor kunci publik Anda dan menghasilkan sumber daya kunci publik.

Untuk mengimpor kunci publik yang ada dengan CLI:

```
aws ivs-realtime import-public-key --public-key-material "`cat public.pem`" --region
<aws-region>
```

Anda dapat menghilangkan `--region <aws-region>` jika wilayah tersebut ada di file konfigurasi AWS lokal Anda.

Berikut adalah contoh respons :

```
{
  "publicKey": {
    "arn": "arn:aws:ivs:us-west-2:123456789012:public-key/f99cde61-c2b0-4df3-8941-
ca7d38acca1a",
    "fingerprint": "98:0d:1a:a0:19:96:1e:ea:0a:0a:2c:9a:42:19:2b:e7",
```

```

    "publicKeyMaterial": "-----BEGIN PUBLIC KEY-----
\nMHYwEAYHKoZIzj0CAQYFK4EEACIDYgAEVjYMV+P4ML6xemanCrtse/FDwsNnpYmS
\nS6vRV9Wx37mjwi02h0bKuCJqpj7x01pz0bHm5v1JBvdZYAd/r2LR5aChK+/GM2Wj
\nl8MG9NJIVFaw1u3bvjEjzTASSfS1BDX1\n-----END PUBLIC KEY-----\n",
    "tags": {}
  }
}

```

Permintaan API

```

POST /ImportPublicKey HTTP/1.1
{
  "publicKeyMaterial": "<pem file contents>"
}

```

Menghasilkan dan Menandatangani Token

Untuk detail tentang bekerja dengan JWTs dan pustaka yang didukung untuk menandatangani token, kunjungi jwt.io. Pada antarmuka jwt.io, Anda harus memasukkan kunci pribadi Anda untuk menandatangani token. Kunci publik hanya diperlukan jika Anda ingin memverifikasi token.

Semua JWTs memiliki tiga bidang: header, payload, dan signature.

Skema JSON untuk header dan payload JWT dijelaskan di bawah ini. Atau Anda dapat menyalin sampel JSON dari konsol IVS. Untuk mendapatkan header dan payload JSON dari konsol IVS:

1. Buka [konsol Amazon IVS](#). Pilih wilayah panggung Anda jika Anda belum berada di sana.
2. Di menu navigasi kiri, pilih Streaming waktu nyata > Tahapan.
3. Pilih tahap yang ingin Anda gunakan. Pilih Lihat detail.
4. Di bagian Token peserta, pilih drop-down di sebelah Buat token.
5. Pilih Build token header dan payload.
6. Isi formulir dan salin header JWT dan payload yang ditunjukkan di bagian bawah popup.

Skema Token: Header

Header menentukan:

- `alg` adalah algoritma penandatanganan. Ini adalah ES384, algoritma tanda tangan ECDSA yang menggunakan algoritma hash SHA-384.

- `typ` adalah jenis token, JWT.
- `kid` adalah ARN dari kunci publik yang digunakan untuk menandatangani token. Itu harus ARN yang sama yang dikembalikan dari permintaan [GetPublicKeyAPI](#).

```
{
  "alg": "ES384",
  "typ": "JWT"
  "kid": "arn:aws:ivs:123456789012:us-east-1:public-key/abcdefg12345"
}
```

Skema Token: Muatan

Muatan berisi data khusus untuk IVS. Semua bidang kecuali `user_id` wajib.

- `RegisteredClaims` dalam spesifikasi JWT adalah klaim cadangan yang perlu disediakan agar token tahap valid:
 - `exp` (waktu kedaluwarsa) adalah stempel waktu Unix UTC saat token kedaluwarsa. (Stempel waktu Unix adalah nilai numerik yang mewakili jumlah detik dari 1970-01-01T 00:00:00 Z UTC hingga tanggal/waktu UTC yang ditentukan, mengabaikan detik kabisat.) Token divalidasi saat peserta bergabung dengan panggung. IVS menyediakan token dengan TTL 12 jam default, yang kami rekomendasikan; ini dapat diperpanjang hingga maksimal 14 hari sejak dikeluarkan pada waktu (`iat`). Ini harus berupa nilai tipe integer.
 - `iat` (dikeluarkan pada waktu) adalah stempel waktu Unix UTC ketika JWT dikeluarkan. (Lihat catatan `exp` tentang stempel waktu Unix.) Ini harus berupa nilai tipe integer.
 - `jti` (JWT ID) adalah ID peserta yang digunakan untuk melacak dan merujuk pada peserta yang diberikan token. Setiap token harus memiliki ID peserta yang unik. Itu harus berupa string case-sensitive, hingga 64 karakter, hanya berisi karakter alfanumerik, tanda hubung (-), dan garis bawah (_). Tidak ada karakter khusus lainnya yang diizinkan.
- `user_id` adalah nama opsional yang ditetapkan pelanggan untuk membantu mengidentifikasi token; ini dapat digunakan untuk menautkan peserta ke pengguna dalam sistem pelanggan sendiri. Ini harus cocok dengan `userId` bidang dalam permintaan [CreateParticipantTokenAPI](#). Ini bisa berupa teks yang dikodekan UTF-8 dan merupakan string hingga 128 karakter. Bidang ini diekspos ke semua peserta tahap dan tidak boleh digunakan untuk mengidentifikasi pribadi, rahasia, atau informasi sensitif.
- `resource` adalah ARN dari panggung; misalnya, `arn:aws:ivs:us-east-1:123456789012:stage/oRmLNwuCeMlQ`

- `topic` adalah ID panggung, yang dapat diekstraksi dari tahap ARN. Misalnya, jika tahap ARN adalah `arn:aws:ivs:us-east-1:123456789012:stage/oRmLNwuCeMlQ`, ID tahap adalah `oRmLNwuCeMlQ`
- `events_url` harus berupa titik akhir peristiwa yang dikembalikan dari `GetStage` operasi `CreateStage` atau. Kami menyarankan Anda menyimpan nilai ini pada waktu pembuatan tahap; nilainya dapat di-cache hingga 14 hari. Contoh nilai adalah `wss://global.events.live-video.net`.
- `whip_url` harus menjadi titik akhir WHIP yang dikembalikan dari operasi `CreateStage` atau `GetStage`. Kami menyarankan Anda menyimpan nilai ini pada waktu pembuatan tahap; nilainya dapat di-cache hingga 14 hari. Contoh nilai adalah `https://453fdfd2ad24df.global-bm.whip.live-video.net`.
- `capabilities` menentukan kemampuan token; nilai yang valid adalah `allow_publish` dan `allow_subscribe`. Untuk token khusus berlangganan, setel hanya ke `allow_subscribe true`
- `attributes` adalah bidang opsional tempat Anda dapat menentukan atribut yang disediakan aplikasi untuk dikodekan ke dalam token dan dilampirkan ke panggung. Kunci dan nilai peta dapat berisi teks yang dikodekan UTF-8. Panjang maksimum bidang ini adalah total 1 KB. Bidang ini diekspos ke semua peserta tahap dan tidak boleh digunakan untuk mengidentifikasi pribadi, rahasia, atau informasi sensitif.
- `version` harus `1.0`.

```
{
  "exp": 1697322063,
  "iat": 1697149263,
  "jti": "Mx6c1RRHODPy",
  "user_id": "<optional_customer_assigned_name>",
  "resource": "<stage_arn>",
  "topic": "<stage_id>",
  "events_url": "wss://global.events.live-video.net",
  "whip_url": "https://114ddfabadaf.global-bm.whip.live-video.net",
  "capabilities": {
    "allow_publish": true,
    "allow_subscribe": true
  },
  "attributes": {
    "optional_field_1": "abcd1234",
    "optional_field_2": "false"
  },
}
```

```
"version": "1.0"
}
```

Skema Token: Tanda Tangan

Untuk membuat tanda tangan, gunakan kunci pribadi dengan algoritme yang ditentukan di header (ES384) untuk menandatangani header yang dikodekan dan muatan yang dikodekan.

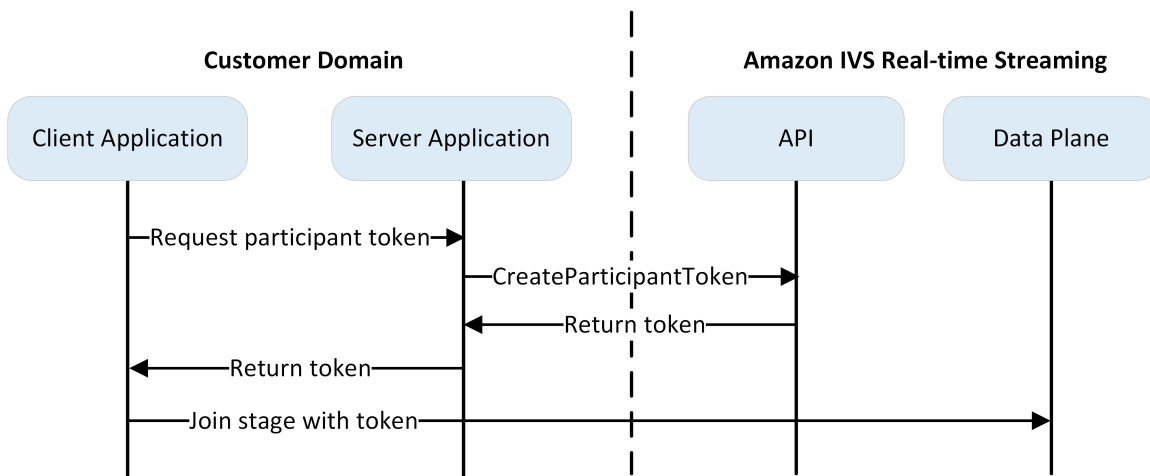
```
ECDSASHA384(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  <private-key>
)
```

Petunjuk

1. Hasilkan tanda tangan token dengan algoritme ES384 penandatanganan dan kunci pribadi yang terkait dengan kunci publik yang diberikan kepada IVS.
2. Merakit token.

```
base64UrlEncode(header) + "." +
base64UrlEncode(payload) + "." +
base64UrlEncode(signature)
```

Membuat Token dengan IVS Real-Time Streaming API



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 menentukan atribut dan/atau kemampuan secara opsional:

- Anda dapat menentukan atribut yang disediakan aplikasi untuk dikodekan ke dalam token dan melampirkan ke panggung. Kunci dan nilai peta dapat berisi teks yang dikodekan UTF-8. Panjang maksimum bidang ini adalah total 1 KB. Bidang ini diekspos ke semua peserta tahap dan tidak boleh digunakan untuk mengidentifikasi pribadi, rahasia, atau informasi sensitif.
- Anda dapat menentukan kemampuan yang diaktifkan oleh token. 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.

Lihat perinciannya di [CreateParticipantToken](#).

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**. Jendela **Create token** muncul.
3. Masukkan ID pengguna untuk dikaitkan dengan token. Ini bisa berupa teks yang dikodekan UTF-8.
4. Pilih **Buat**.
5. Salin token. **Penting:** Pastikan untuk menyimpan token; IVS tidak menyimpannya dan Anda tidak dapat mengambilnya nanti.

Prasyarat: Untuk menggunakan contoh kode di bawah ini, Anda perlu menginstal paket `aws-sdk/client-ivs-realtime`. Untuk detailnya, lihat [Memulai AWS SDK for JavaScript](#).

```
import { IVSRealTimeClient, CreateParticipantTokenCommand } from "@aws-sdk/client-ivs-realtime";

const ivsRealtimeClient = new IVSRealTimeClient({ region: 'us-west-2' });
const stageArn = 'arn:aws:ivs:us-west-2:123456789012:stage/L210UYabcdef';
const createStageTokenRequest = new CreateParticipantTokenCommand({
  stageArn,
});
const response = await ivsRealtimeClient.send(createStageTokenRequest);
console.log('token', response.participantToken.token);
```

Langkah 4: Integrasikan SDK Siaran IVS

IVS menyediakan SDK siaran untuk web, Android, dan iOS yang dapat Anda integrasikan ke dalam aplikasi Anda. SDK siaran digunakan untuk mengirim dan menerima video. Jika Anda telah [mengonfigurasi RTMP Ingest untuk tahap Anda](#), Anda dapat menggunakan encoder apa pun yang dapat disiarkan ke titik akhir RTMP (misalnya, OBS atau ffmpeg).

Pada bagian ini, kami menulis aplikasi sederhana yang memungkinkan dua atau lebih peserta untuk berinteraksi secara real time. Langkah-langkah di bawah ini memandu Anda melalui pembuatan aplikasi yang disebut BasicRealTime. Kode aplikasi lengkap aktif CodePen dan GitHub:

- Situs web: <https://codepen.io/amazon-ivs/pen/ZEggrpo>
- Android: <https://github.com/aws-samples/amazon-ivs-real-time-streaming-android-samples>
- 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 Script

SDK siaran Web didistribusikan sebagai JavaScript perpustakaan dan dapat diambil di <https://web-broadcast.live-video.net/1.33.0/amazon-ivs-web-broadcast.js>.

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

Menggunakan npm

Untuk menginstal paket npm:

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

Anda sekarang dapat mengakses objek `IVSBroadcastClient`:

```
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 APIs (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, mavenCentral1 repositori mungkin sudah disertakan dalam settings.gradle file Anda, jika itu masalahnya Anda dapat menghilangkan bloknnya. 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:1.40.0: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. Untuk petunjuknya, lihat [Menginstal Pustaka](#) di Panduan SDK Siaran iOS.

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

Langkah 5: Publikasikan dan Berlangganan Video

Anda dapat publish/subscribe (real-time) ke IVS dengan:

- [Siaran IVS](#) asli SDKs, yang mendukung WebRTC dan RTMPS. Kami merekomendasikan ini, terutama untuk skenario produksi. Lihat detail di bawah ini untuk [Web](#), [Android](#), dan [iOS](#).
- Konsol Amazon IVS — Ini cocok untuk menguji aliran. Lihat di bawah ini.
- Perangkat lunak streaming dan encoder perangkat keras lainnya - Anda dapat menggunakan encoder streaming apa pun yang mendukung protokol RTMP, RTMPS, atau WHIP. Lihat [Stream Ingest](#) untuk informasi lebih lanjut.

Konsol IVS

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

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

2. Di panel navigasi, pilih Tahapan. (Jika panel navigasi dicituk, perluas dengan memilih ikon hamburger.)
3. Pilih tahap yang ingin Anda berlangganan atau terbitkan, untuk pergi ke halaman detailnya.
4. Untuk berlangganan: Jika panggung memiliki satu atau lebih penerbit, Anda dapat berlangganan dengan menekan tombol Berlangganan, di bawah tab Berlangganan. (Tab berada di bawah bagian Konfigurasi Umum.)
5. Untuk mempublikasikan:
 - a. Pilih tab Publikasikan.
 - b. Anda akan diminta untuk memberikan akses konsol IVS ke kamera dan mikrofon Anda; Izinkan izin tersebut.

- c. Di bagian bawah tab Publish, gunakan kotak dropdown untuk memilih perangkat input untuk mikrofon dan kamera.
- d. Untuk memulai penerbitan, pilih Mulai penerbitan.
- e. Untuk melihat konten yang dipublikasikan, kembali ke tab Berlangganan.
- f. Untuk berhenti menerbitkan, buka tab Publish dan tekan tombol Stop publishing di bagian bawah.

Catatan: Berlangganan dan penerbitan menghabiskan sumber daya, dan Anda akan dikenakan tarif per jam untuk waktu Anda terhubung ke panggung. Untuk mempelajari lebih lanjut, lihat [Streaming Waktu Nyata](#) di halaman Harga IVS.

Publikasikan & Berlangganan dengan IVS Web Broadcast SDK

Bagian ini akan membawa Anda melalui langkah-langkah yang terlibat dalam penerbitan dan berlangganan ke panggung menggunakan aplikasi web Anda.

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/1.33.0/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 Join/Leave Tombol

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?

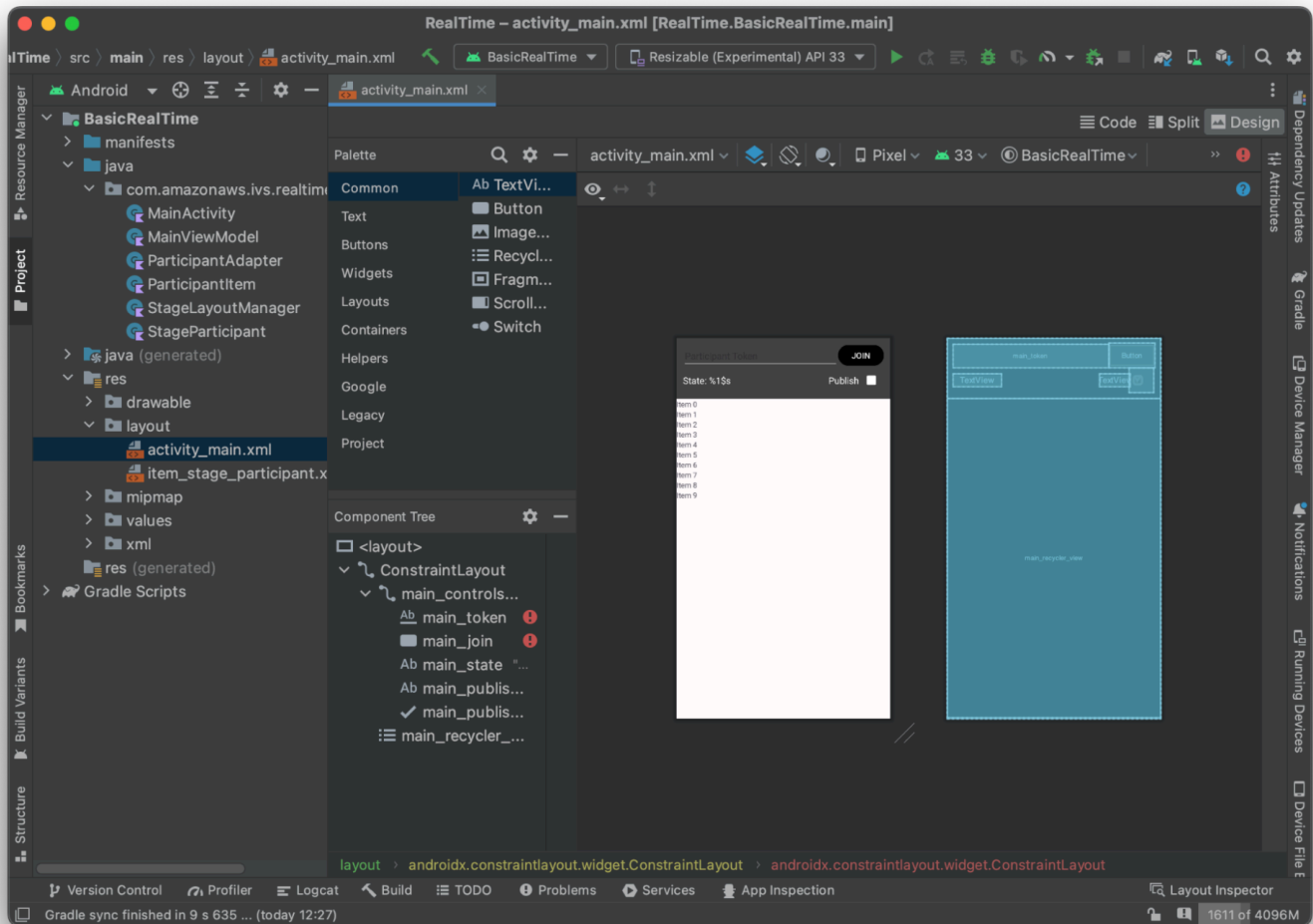
Untuk contoh lebih detail yang melibatkan npm, React, dan lainnya, lihat [IVS Broadcast SDK: Web Guide \(Real-Time Streaming Guide\)](#).

Publikasikan & Berlangganan dengan IVS Android Broadcast SDK

Bagian ini akan membawa Anda melalui langkah-langkah yang terlibat dalam menerbitkan dan berlangganan ke panggung menggunakan aplikasi Android Anda.

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 merferensikan beberapa string IDs 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

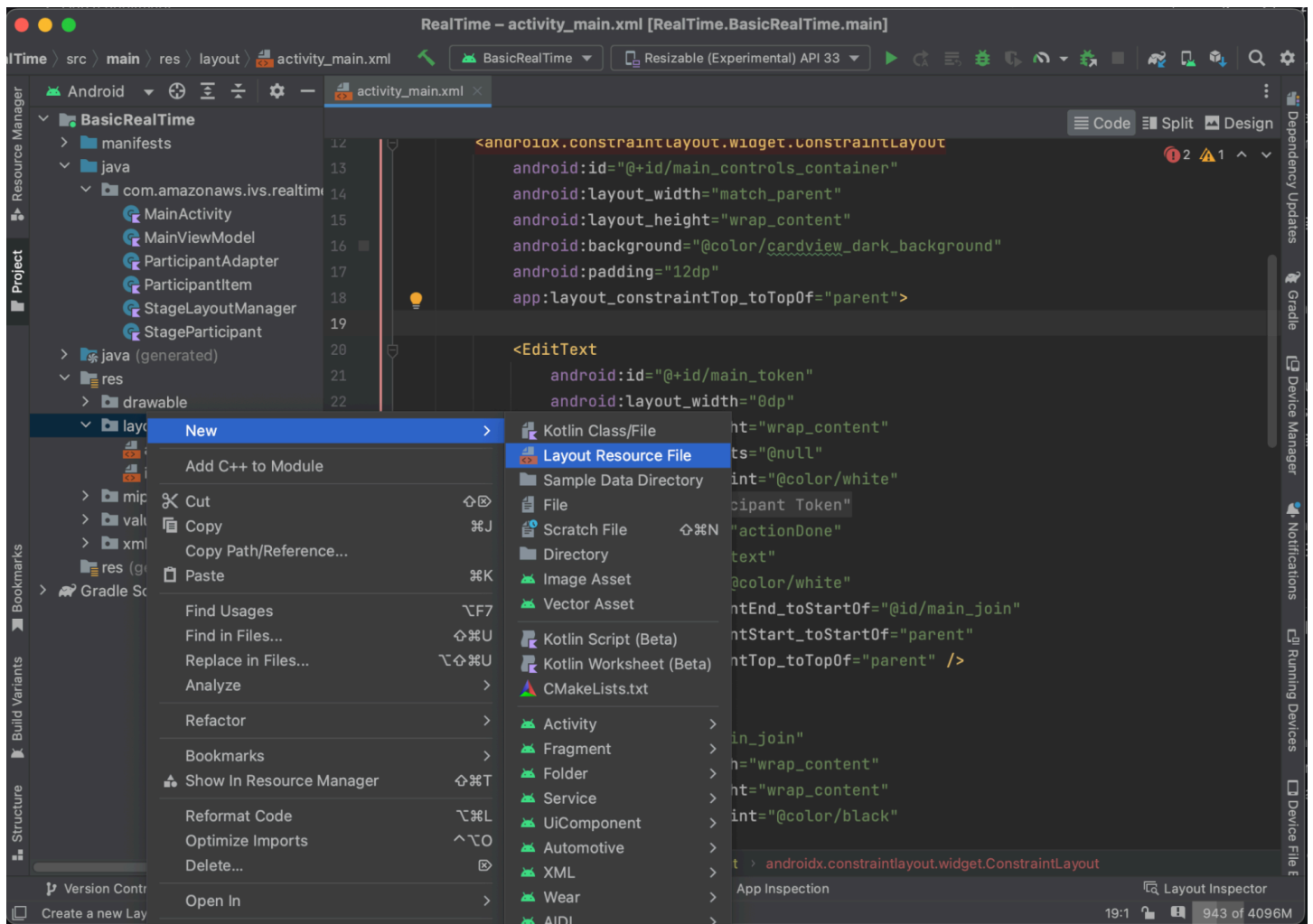
```
import android.widget.Button
import android.widget.CheckBox
import android.widget.EditText
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView

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

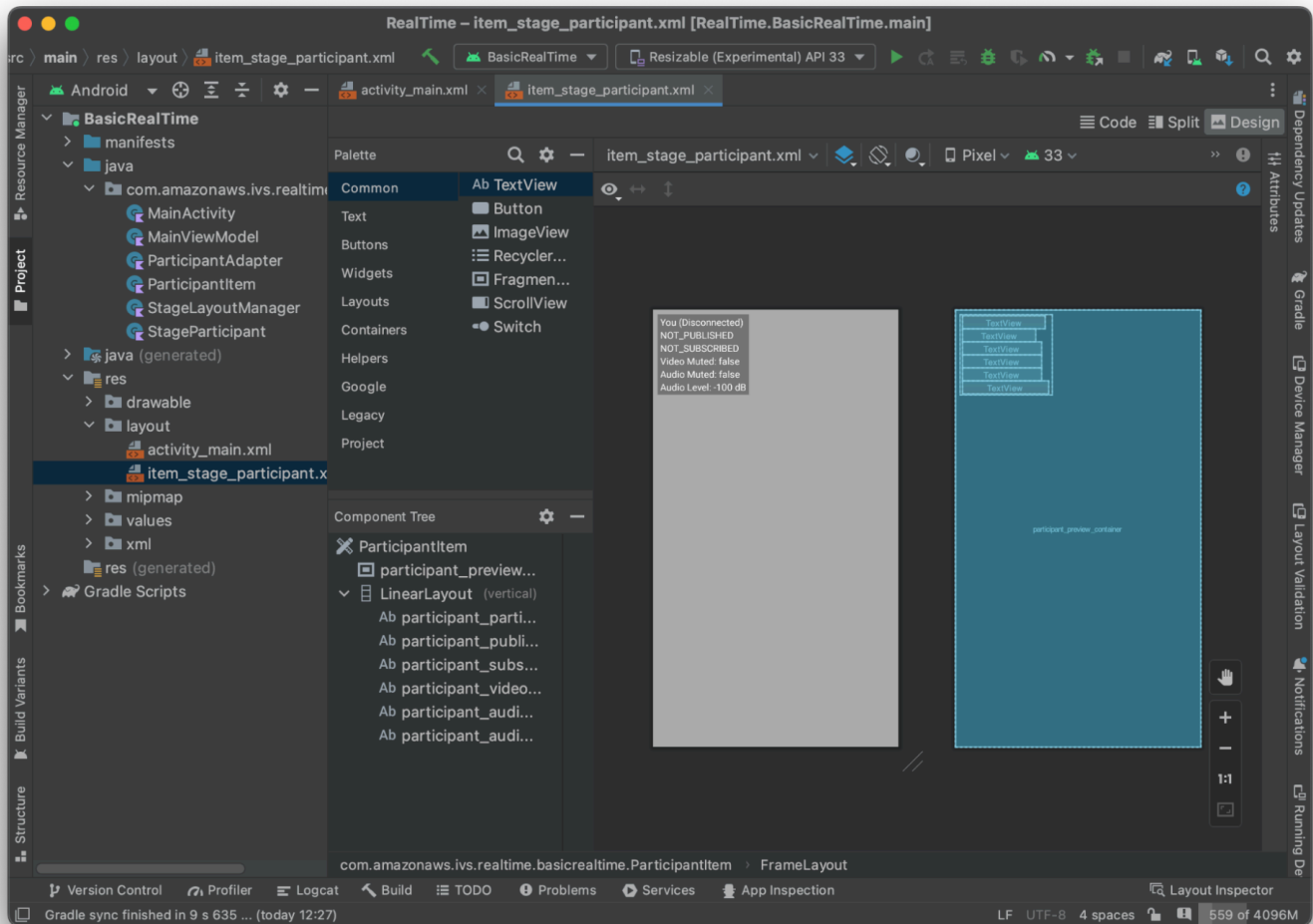
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    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 kita RecyclerView. Untuk melakukan ini, klik kanan res/layout direktori Anda dan pilih New > Layout Resource File. Beri nama file baru ini item_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 row/column tata letak 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.addRender(er(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 diredam, kita perlu menemukan ImageDevice dan AudioDevice dari array stream. Untuk mengoptimalkan kinerja, kami mengingat perangkat terakhir yang terpasang IDs.

```
// 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
```

Publikasikan & Berlangganan dengan IVS iOS Broadcast SDK

Bagian ini akan membawa Anda melalui langkah-langkah yang terlibat dalam penerbitan dan berlangganan ke panggung menggunakan aplikasi iOS Anda.

Buat Tampilan

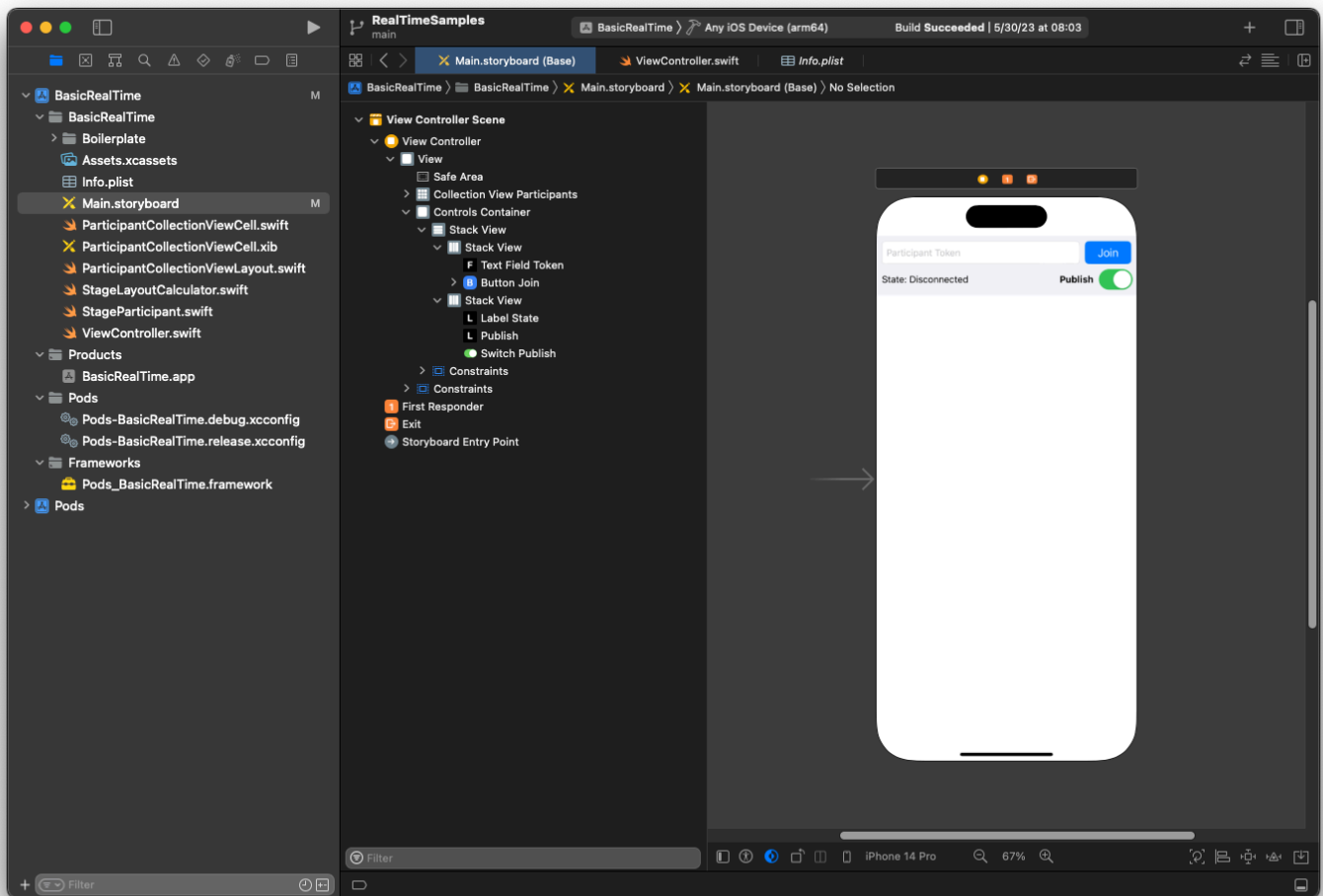
Kami mulai dengan menggunakan `ViewController.swift` file yang dibuat secara otomatis untuk mengimpor `AmazonIVSBroadcast` dan kemudian menambahkan beberapa `@IBOutlet` ke tautan:

```
import AmazonIVSBroadcast

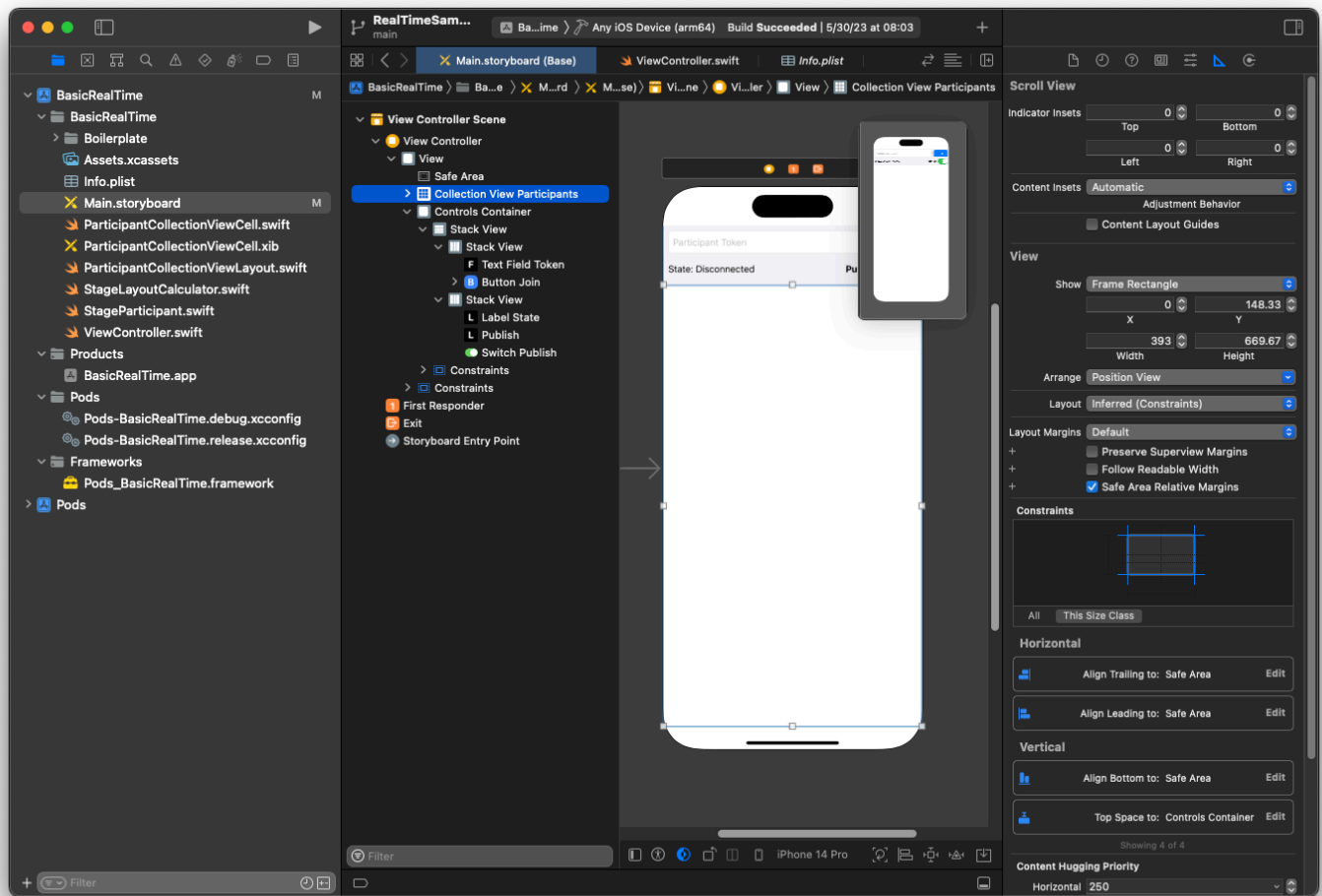
class ViewController: UIViewController {

    @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!
```

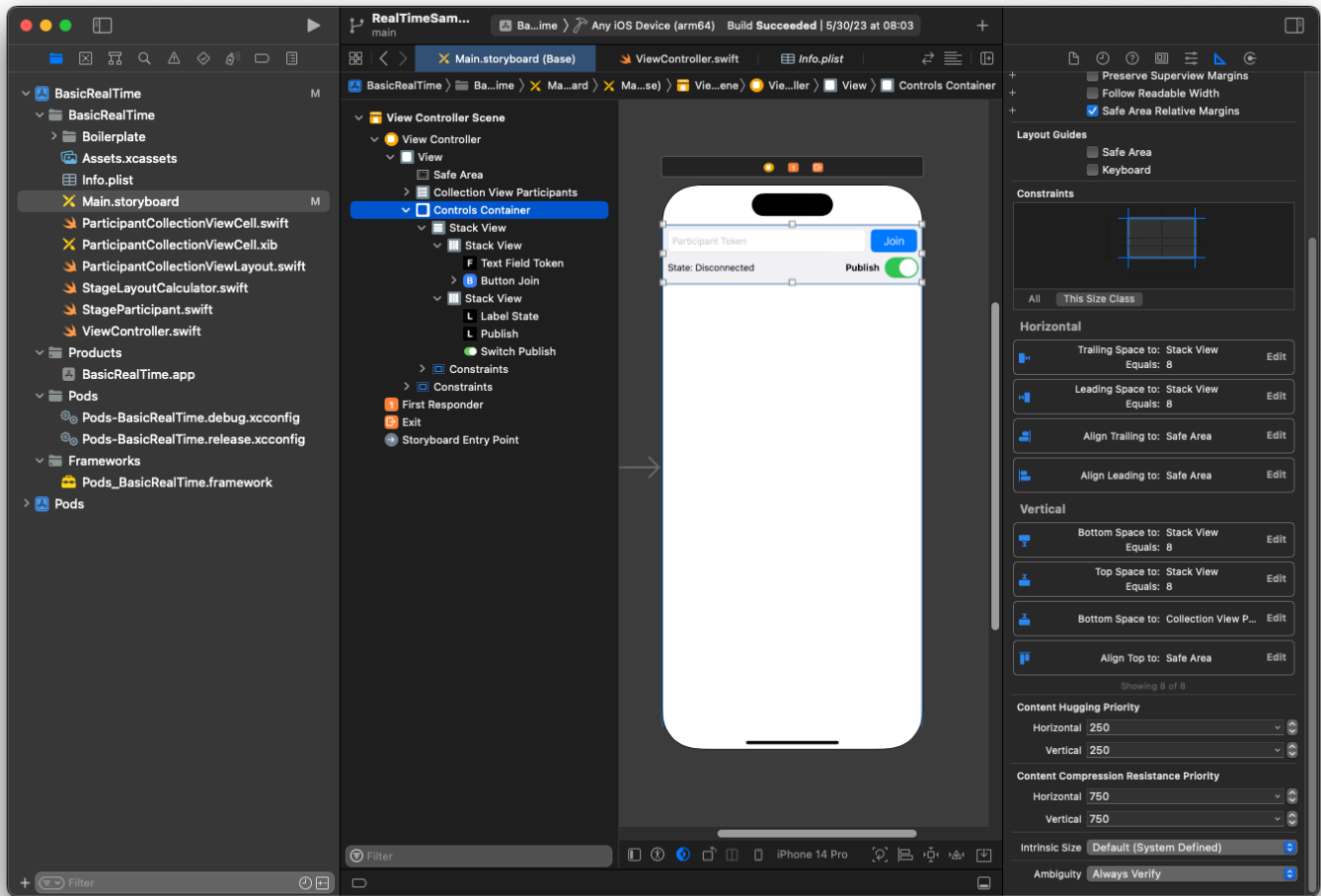
Sekarang kita membuat pandangan itu dan menghubungkannya `Main.storyboard`. Berikut adalah struktur tampilan yang akan kita gunakan:



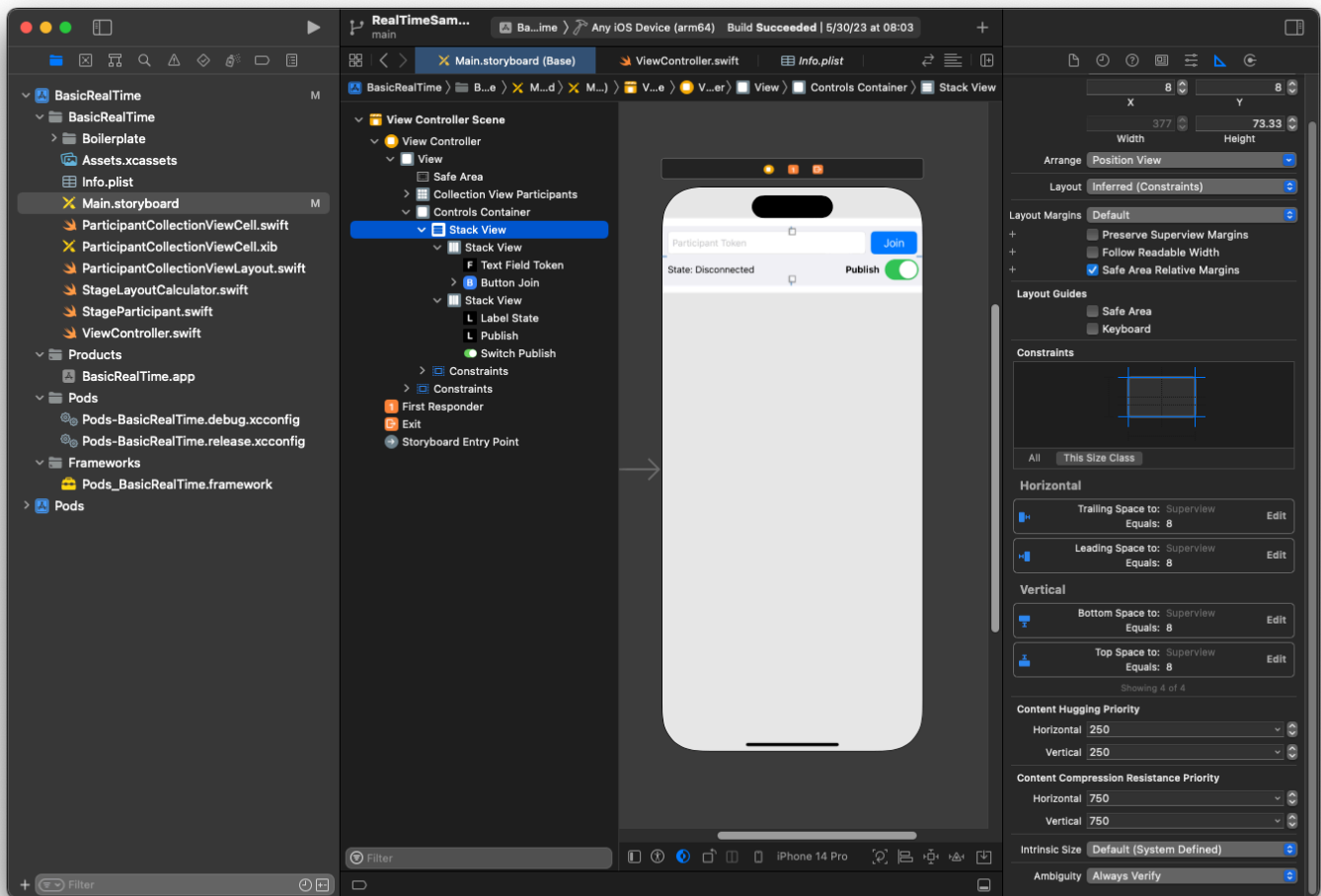
Untuk AutoLayout konfigurasi, kita perlu menyesuaikan tiga tampilan. Tampilan pertama adalah Collection View Partisipan (`UICollectionView`). Terikat Memimpin, Tertinggal, dan Bawah ke Area Aman. Juga terikat Atas ke Controls Container.



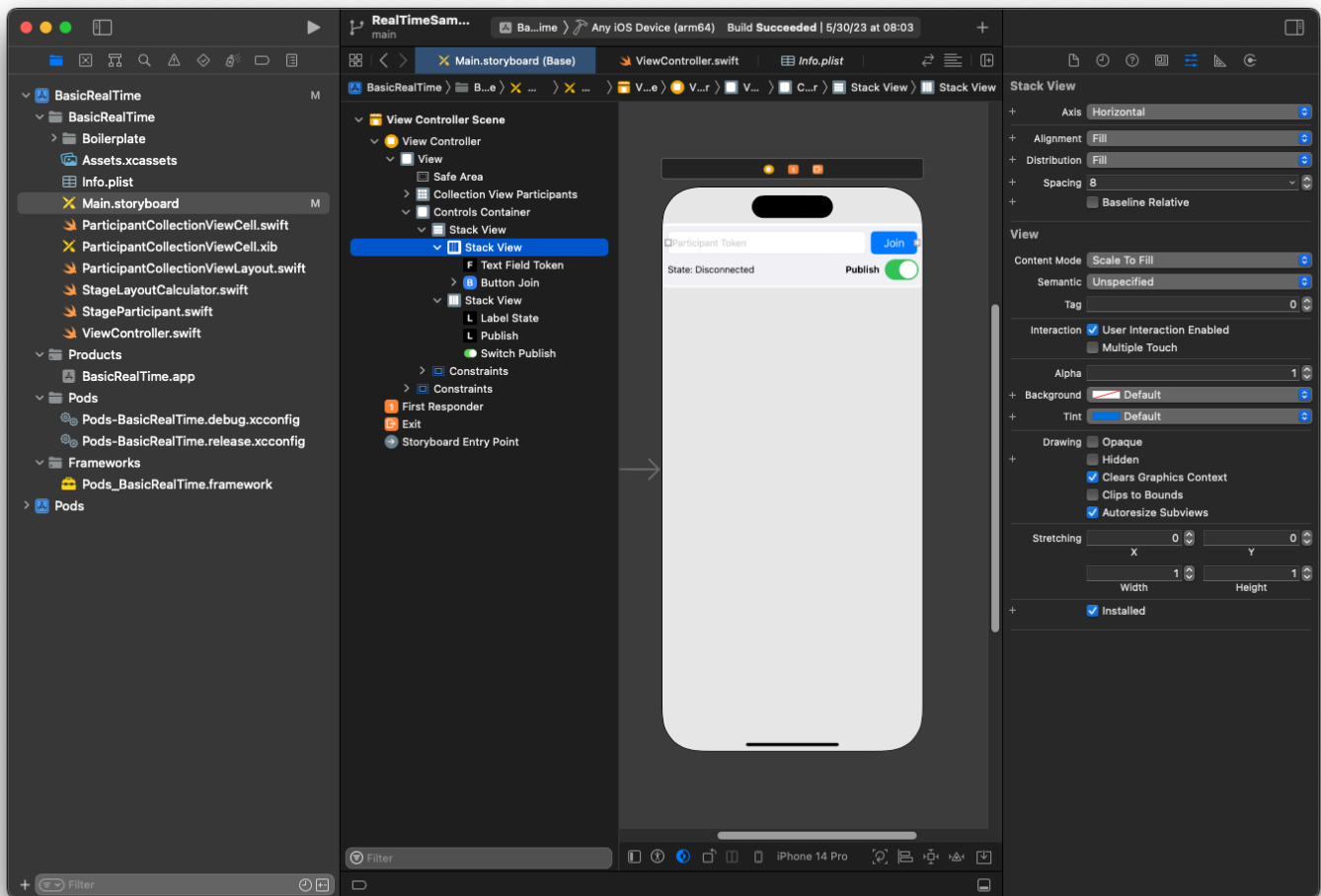
Tampilan kedua adalah Controls Container. Terikat Memimpin, Melintasi, dan Atas ke Area Aman:



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



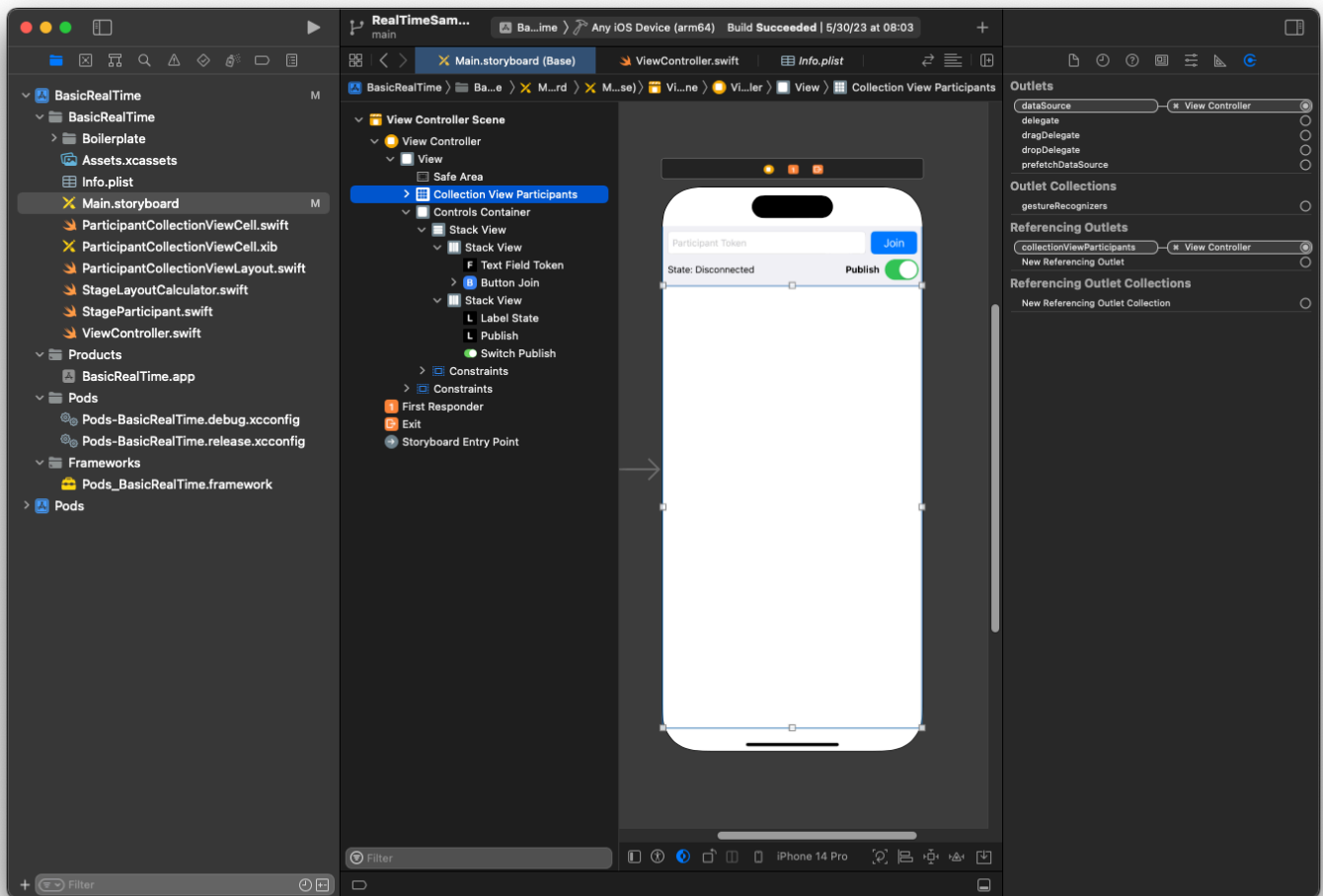
UIStackTampilan akan menangani tata letak tampilan yang tersisa. Untuk ketiga UIStackTampilan, gunakan Fill sebagai Alignment dan Distribution.



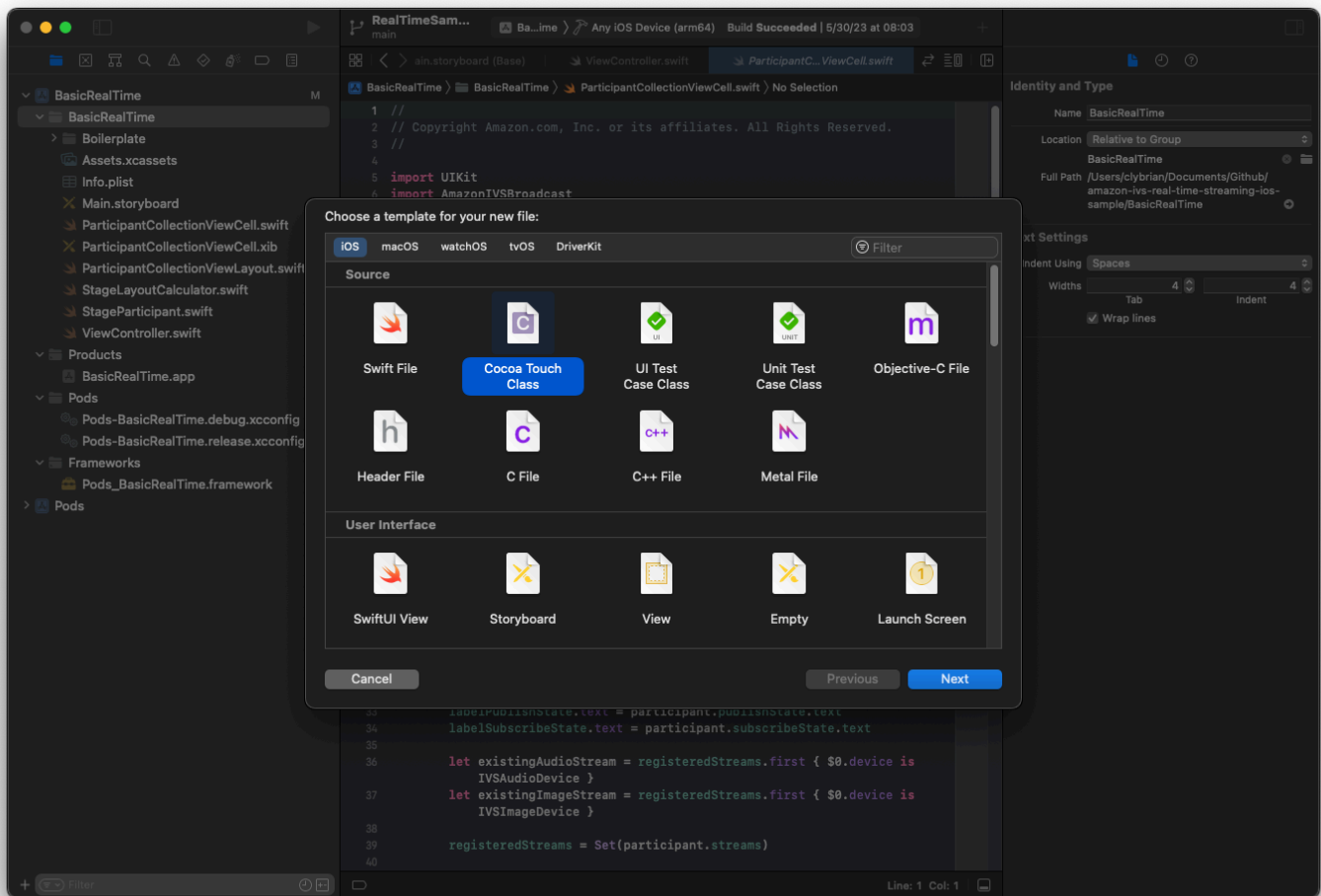
Akhirnya, mari kita tautkan pandangan ini ke pandangan kita `ViewController`. Dari atas, petakan tampilan berikut:

- 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 menyetel item Peserta Tampilan Koleksi menjadi milik: `dataSource ViewController`



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



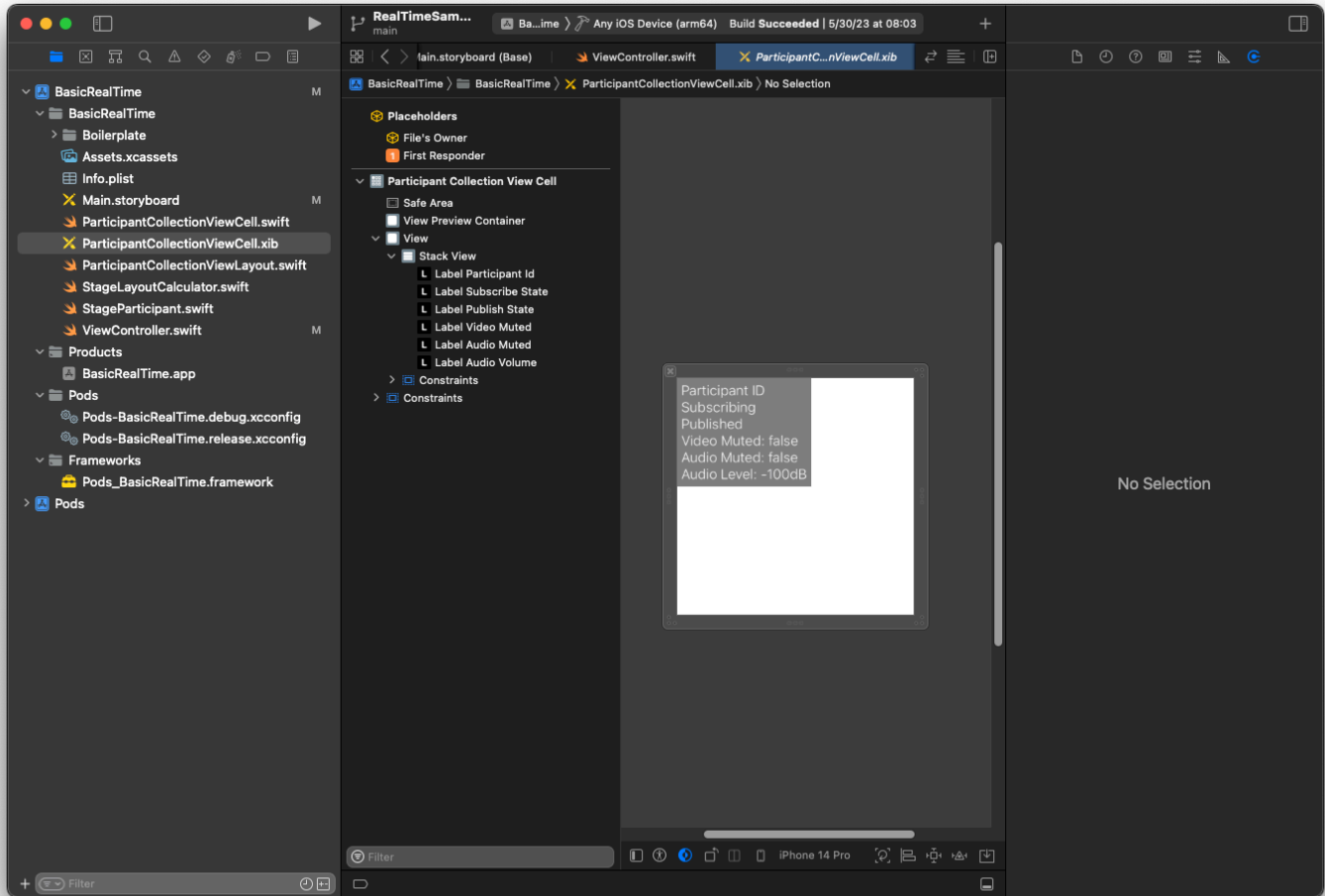
Beri nama `ParticipantUICollectionViewCell` dan jadikan subclass `UICollectionViewCell` di Swift. Kami mulai di file Swift lagi, membuat tautan `@IBOutlet` untuk kami:

```
import AmazonIVSBroadcast

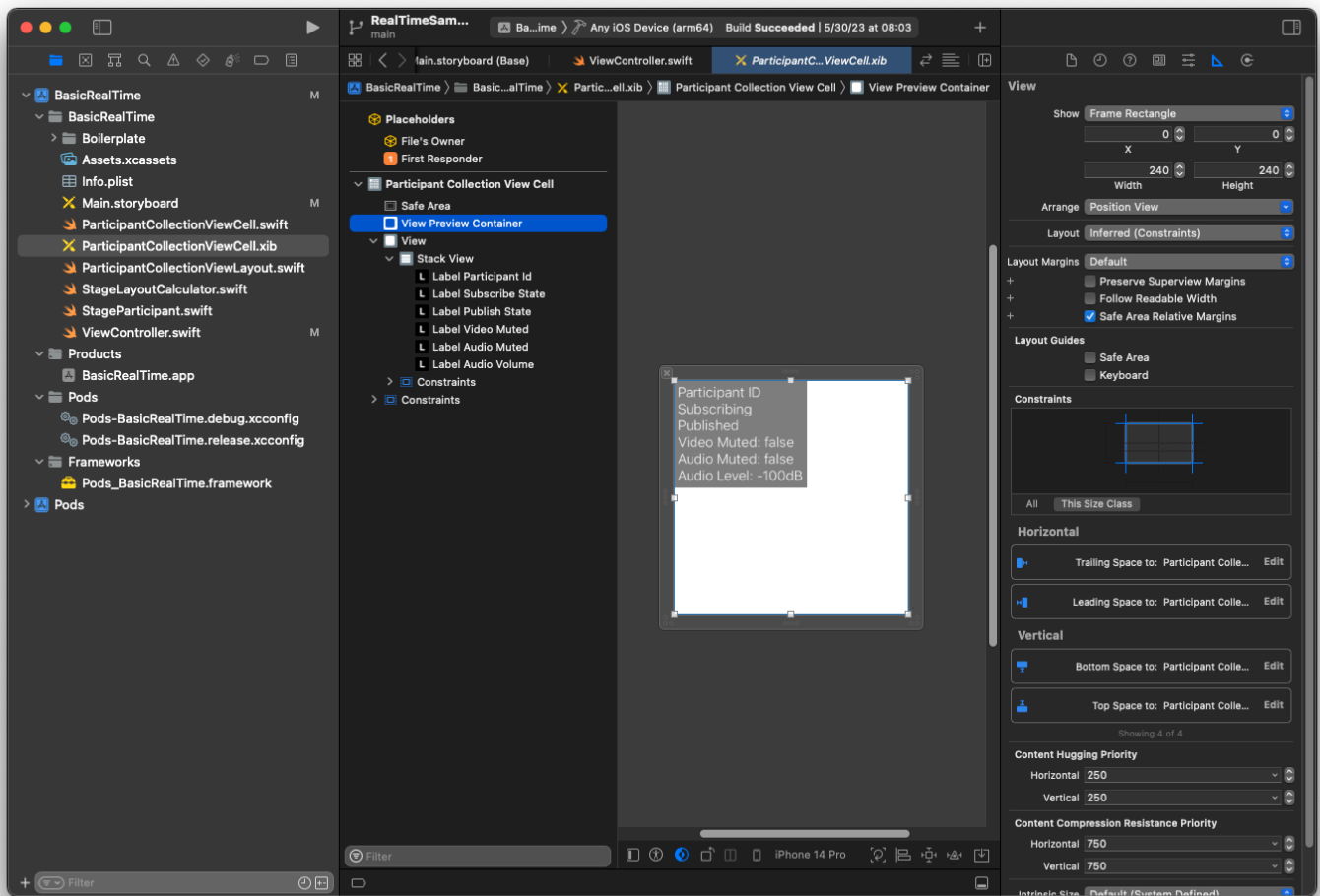
class ParticipantCollectionViewCell: UICollectionViewCell {

    @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!
```

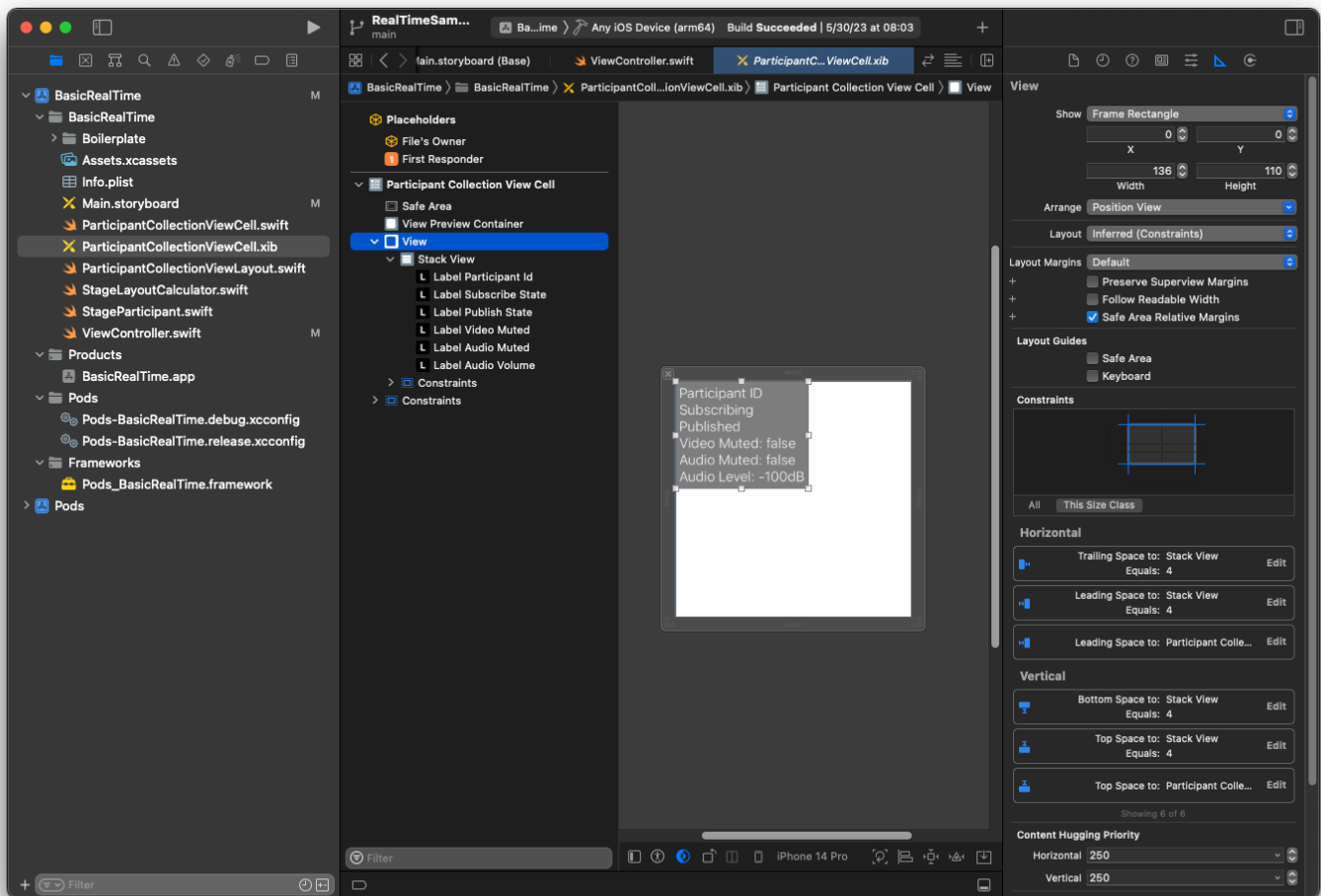
Dalam file XIB terkait, buat hierarki tampilan ini:



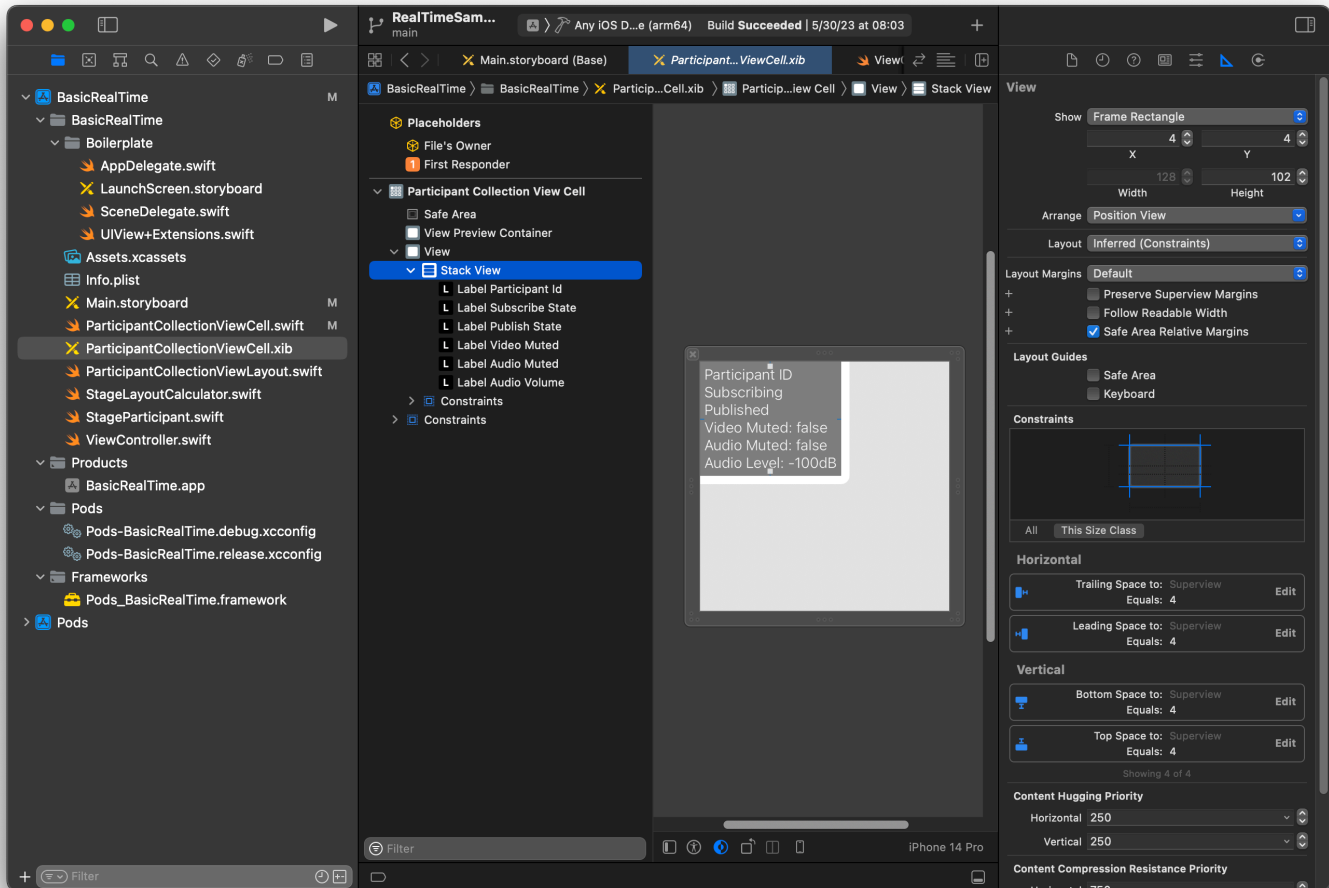
Untuk AutoLayout, kita akan memodifikasi tiga tampilan lagi. Tampilan pertama adalah View Preview Container. Atur Trailing, Leading, Top, dan Bottom ke Sel Tampilan Koleksi Peserta.



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 reloadItems, 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.

IVSStageStrategi

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.

IVSStagePenyaji

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 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 Kustom UICollectionView ViewLayout

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 `aUICollectionViewLayout`.

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 row/column tata letak 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)

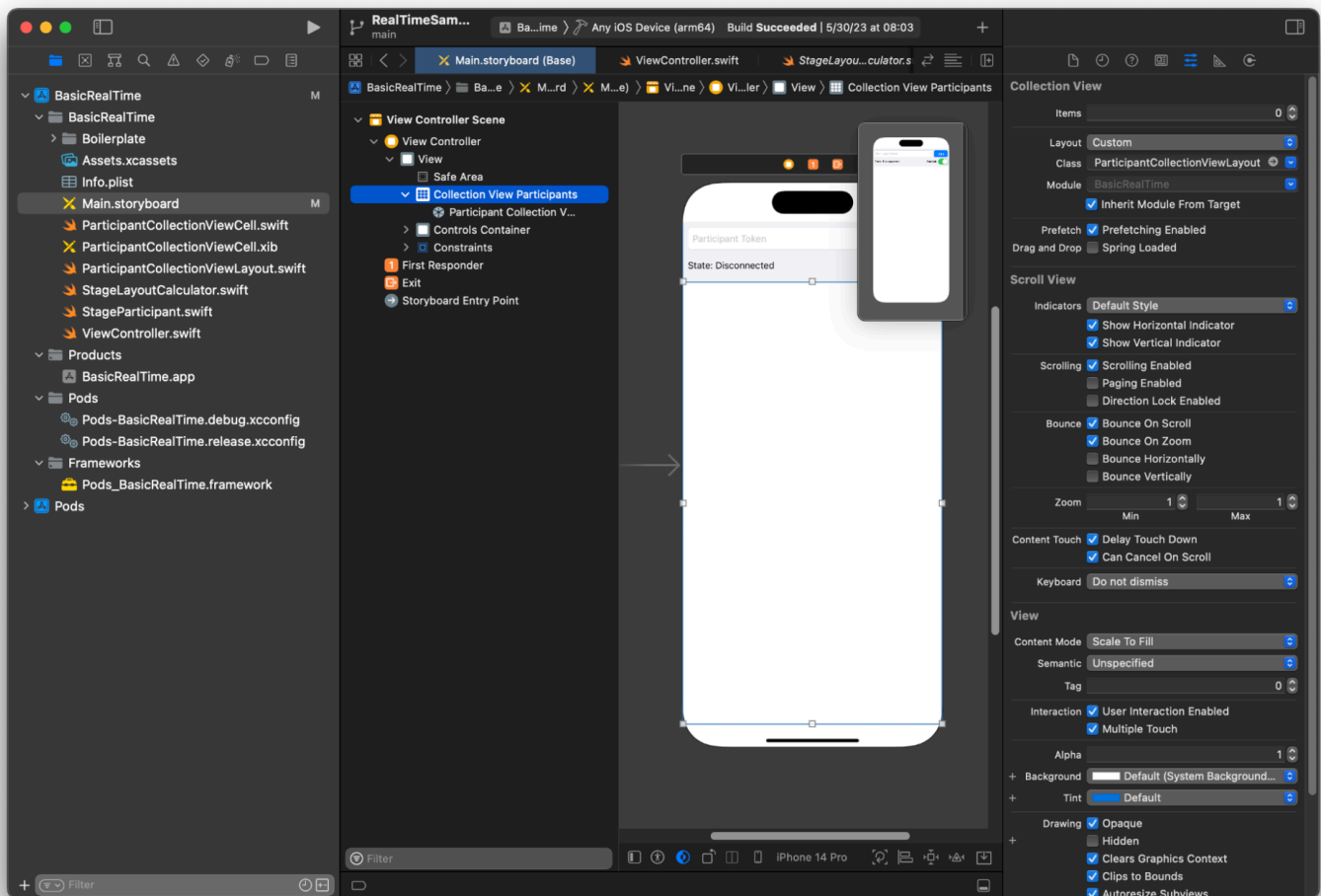
    var frames = [CGRect]()
    for row in 0 ..< layout.count {
        // layout[row] is the number of columns in a layout
        let itemWidth = (isVertical ? width : height) / CGFloat(layout[row])
        let segmentFrame = CGRect(x: (isVertical ? 0 : lastFrame.maxX) +
halfPadding,
                 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

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

    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

Dokumen ini memberikan detail tentang opsi yang tersedia untuk memantau aplikasi streaming real-time IVS Anda.

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 [siaran SDKs 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"
  }
]
}
```

CloudWatch Metrik 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 menu tarik-turun Metrik, lalu pilih Semua metrik.
3. Pada tab Jelajahi, menggunakan menu tarik-turun tidak berlabel di sebelah kiri, pilih wilayah “asal” Anda, tempat saluran dibuat. Untuk informasi selengkapnya 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 ruang nama IVSRealTime.
5. Lakukan salah satu tindakan berikut:
 - a. Di bilah pencarian, masukkan ID sumber daya Anda (bagian dari ARN, `arn::ivs:stage/<resource id>`).

Kemudian pilih IVSRealTime > Stage Metrics.

 - b. Jika IVSRealWaktu muncul sebagai layanan yang dapat dipilih di bawah AWS Namespaces, pilih. Ini akan terdaftar jika Anda menggunakan Amazon IVS Real-Time Streaming dan

mengirimkan metrik ke Amazon. CloudWatch (Jika IVSRealWaktu 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 yang akan 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 dengan memanfaatkan AWS CLI. Cara ini mengharuskan Anda untuk mengunduh dan mengonfigurasi CLI terlebih dahulu di mesin. Untuk detail, lihat [Panduan Pengguna AWS Command Line Interface](#).

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

- Di 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 ruang nama AWS/Time IVSReal.

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 “none” untuk “video”, atau “none” untuk “audio.”
Media Type Nilai ini juga bisa kosong.

- `MediaTypeDimensinya` adalah “video” atau “audio” (string).

Dalam hal replikasi peserta, untuk tahap tujuan, metrik tahap-kesehatan yang ada mencakup semua peserta yang direplikasi (penerbit di tahap sumber yang merupakan peserta replika di tahap tujuan).

Metrik	Dimensi	Deskripsi
<code>ConcurrentPublishers</code>	—	<p>Jumlah peserta yang mempublikasikan di semua tahapan di Wilayah AWS.</p> <p>Unit: Jumlah</p> <p>Statistik yang valid: Rata-rata, Maksimum, Minimum</p>
<code>ConcurrentSubscriptions</code>	—	<p>Jumlah publisher-to-subscriber koneksi simultan di semua tahapan di Wilayah AWS.</p> <p>Unit: Jumlah</p> <p>Statistik yang valid: Rata-rata, Maksimum, Minimum</p>
<code>DownloadPacketLoss</code>	—	<p>Persentase paket yang hilang saat mengunduh dari server IVS oleh pelanggan.</p> <p>Unit: 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>
<code>DownloadPacketLoss</code>	<code>Platform</code>	<p>Filter <code>DownloadPacketLoss</code> berdasarkan platform pelanggan.</p> <p>Unit: 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>

Metrik	Dimensi	Deskripsi
DownloadPacketLoss	Platform, SDKVersion	<p>Filter DownloadPacketLoss berdasarkan platform pelanggan dan versi SDK.</p> <p>Unit: 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	<p>Filter DownloadPacketLoss berdasarkan tahap pelanggan.</p> <p>Unit: 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>Unit: 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>

Metrik	Dimensi	Deskripsi
DownloadPacketLoss	Stage, Platform	<p>Filter DownloadPacketLoss berdasarkan tahap pelanggan dan platform.</p> <p>Unit: 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, Platform, SDKVersion	<p>Filter DownloadPacketLoss berdasarkan tahap pelanggan, platform, dan versi SDK.</p> <p>Unit: 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, SubscriberCountryCode	<p>Filter DownloadPacketLoss berdasarkan tahap pelanggan dan kode negara (ISO 3166).</p> <p>Unit: 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>

Metrik	Dimensi	Deskripsi
DownloadPacketLoss	SubscriberCountryCode	<p>Filter DownloadPacketLoss berdasarkan kode negara pelanggan (ISO 3166).</p> <p>Unit: 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>
DroppedFrames	—	<p>Untuk pelanggan: persentase bingkai video turun, dihitung dengan menggabungkan bingkai yang diterima dan bingkai yang dijatuhkan di semua penerbit tempat pelanggan berlangganan.</p> <p>Unit: 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	Platform	<p>Filter DroppedFrames berdasarkan platform pelanggan.</p> <p>Unit: 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>

Metrik	Dimensi	Deskripsi
DroppedFrames	Platform, SDKVersion	<p>Filter DroppedFrames berdasarkan platform pelanggan dan versi SDK.</p> <p>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>Filter DroppedFrames berdasarkan tahap.</p> <p>Unit: 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 berdasarkan tahap dan peserta. Hanya dipancarkan untuk pelanggan yang juga penerbit.</p> <p>Unit: 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, Platform	<p>Filter DroppedFrames berdasarkan panggung dan platform pelanggan.</p> <p>Unit: 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>

Metrik	Dimensi	Deskripsi
DroppedFrames	Stage, Platform, SDKVersion	<p>Filter DroppedFrames berdasarkan tahap dan platform pelanggan dan versi SDK.</p> <p>Unit: 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, SubscriberCountryCode	<p>Filter DroppedFrames berdasarkan tahap dan negara pelanggan.</p> <p>Unit: 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	SubscriberCountryCode	<p>Filter DroppedFrames menurut negara pelanggan.</p> <p>Unit: 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>

Metrik	Dimensi	Deskripsi
PublishBitrate	–	<p>Tingkat total di mana penerbit mengirim data video dan audio (dijumlahkan di semua lapisan simulcast). Ini termasuk data yang ditransmisikan kembali. Bitrate dapat meningkat dengan kehilangan paket unggah dan transmisi ulang, karena mencerminkan apa yang dikirim penerbit dan mungkin tidak cocok dengan apa yang diterima atau dikirim IVS kepada pelanggan.</p> <p>Bit/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	Platform	<p>Filter PublishBitrate berdasarkan platform penerbit.</p> <p>Bit/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	<p>Filter PublishBitrate berdasarkan tahap.</p> <p>Satuan: Bits/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
PublishBitrate	Stage, Participant, Simulcast Layer, MediaType	<p>Filter PublishBitrate berdasarkan tahap, peserta, lapisan simulcast, dan jenis media. ID lapisan simulcast diatur oleh SDK siaran. Ketika simulcast dinonaktifkan, ID lapisan ini akan diatur ke “dininaktifkan”. Jenis media adalah “video” atau “audio”.</p> <p>Satuan: Bits/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>
Publishers	Stage	<p>Jumlah peserta yang mempublikasikan ke panggung.</p> <p>Unit: Jumlah</p> <p>Statistik yang valid: Rata-rata, Maksimum, Minimum</p>
PublishFramerate	Stage, Participant	<p>Seberapa sering bingkai video diterima dari penerbit tertentu. Metrik ini hanya tersedia untuk peserta yang menerbitkan melalui RTMP.</p> <p>Satuan: Hitung/detik</p> <p>Statistik yang valid: Rata-rata, Maksimum, Minimum — Jumlah rata-rata, jumlah terbesar, atau angka terkecil (masing-masing) framerate selama interval yang dikonfigurasi</p>

Metrik	Dimensi	Deskripsi
PublishFrameRate	Stage, Participant, Simulcast Layer, MediaType	<p>Seberapa sering bingkai video diterima dari penerbit tertentu. Metrik ini hanya tersedia untuk peserta yang menerbitkan melalui RTMP.</p> <p>Satuan: Hitung/detik</p> <p>Statistik yang valid: Rata-rata, Maksimum, Minimum — Jumlah rata-rata, jumlah terbesar, atau angka terkecil (masing-masing) framerate selama interval yang dikonfigurasi</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	—	<p>Tingkat total di mana pelanggan menerima data video dan audio</p> <p>Satuan: Bits/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
Subscribe Bitrate	Platform	<p>Filter <code>SubscribeBitrate</code> berdasarkan platform pelanggan.</p> <p>Satuan: Bits/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>
Subscribe Bitrate	Platform, SDKVersion	<p>Filter <code>SubscribeBitrate</code> berdasarkan platform pelanggan dan versi SDK.</p> <p>Satuan: Bits/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>
Subscribe Bitrate	Stage	<p>Filter <code>SubscribeBitrate</code> berdasarkan tahap.</p> <p>Satuan: Bits/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
Subscribe Bitrate	Stage, Participant, MediaType	<p>Filter <code>SubscribeBitrate</code> berdasarkan tahap, peserta, dan jenis media. Jenis media adalah “video” atau “audio”. Metrik ini dipancarkan hanya saat peserta berlangganan juga menerbitkan.</p> <p>Satuan: Bits/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>
Subscribe Bitrate	Stage, Platform	<p>Filter <code>SubscribeBitrate</code> berdasarkan panggung dan platform pelanggan.</p> <p>Bit/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>
Subscribe Bitrate	Stage, Platform, SDKVersion	<p>Filter <code>SubscribeBitrate</code> berdasarkan tahap dan platform pelanggan dan versi SDK.</p> <p>Bit/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
Subscribe Bitrate	Stage, Subscribe rCountryCode	<p>Filter <code>SubscribeBitrate</code> berdasarkan tahap dan kode negara pelanggan.</p> <p>Bit/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>
Subscribe Bitrate	Subscribe rCountryCode	<p>Filter <code>SubscribeBitrate</code> berdasarkan kode negara pelanggan (ISO 3166-1 alpha-2).</p> <p>Bit/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>
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 Native

Platform	Versi yang Didukung
Android	9.0+ -- perhatikan pelanggan dapat membangun dengan versi 6.0+ tetapi tidak akan dapat menggunakan fungsionalitas streaming real-time.
iOS	14+

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.

Peramban Desktop

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

Browser Seluler (iOS dan Android)

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

Keterbatasan yang Sudah Diketahui

- Di semua browser web seluler, kami merekomendasikan publishing/subscribing dengan tidak lebih dari tiga penerbit simultan, karena kendala kinerja yang menyebabkan artefak video dan layar hitam. Jika Anda memerlukan lebih banyak penerbit, 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 (TVs, 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.

Support

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.

Penentuan versi

Siaran Amazon IVS SDKs 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 versi sebelumnya ditambahkan sebagai rilis minor dari versi terbaru. Dalam hal ini, rangkaian fitur baru berikutnya akan ditambahkan sebagai versi 4.2.0.

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

Perbaikan bug besar yang kompatibel dengan versi sebelumnya ditangani secara berbeda; perbaikan ini ditambahkan ke beberapa versi:

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

Perbaikan bug besar ditentukan oleh tim produk Amazon IVS. Contoh umumnya adalah pembaruan keamanan yang penting dan perbaikan pilihan lainnya yang diperlukan 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 beberapa nomor patch mungkin tetap bersifat 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: [1.33.0 \(Catatan Rilis\)](#)

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-referensi>](#). Pastikan versi SDK terbaru dipilih.

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

- [Pemutaran Sederhana](#)
- [Penerbitan dan Berlangganan Sederhana](#)
- [Demo Kolaborasi Real-Time React Komprehensif](#)

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

Catatan: Menerbitkan dari browser nyaman bagi pengguna akhir karena tidak memerlukan menginstal perangkat lunak tambahan. Namun, penerbitan berbasis browser tunduk pada kendala dan variabilitas lingkungan browser. Jika Anda perlu memprioritaskan stabilitas (misalnya, untuk streaming acara), kami umumnya merekomendasikan penerbitan dari sumber non-browser (misalnya, OBS Studio atau encoder khusus lainnya), yang sering memiliki akses langsung ke

sumber daya sistem dan menghindari keterbatasan browser. Untuk informasi lebih lanjut tentang opsi penerbitan non-browser, lihat dokumentasi [Stream Ingest](#).

Memulai dengan SDK Siaran Web IVS | Streaming Waktu Nyata

Dokumen ini membawa Anda melalui langkah-langkah yang terlibat dalam memulai dengan SDK siaran Web streaming real-time IVS.

Impor

Blok bangunan untuk waktu nyata terletak di namespace yang berbeda dari modul penyiaran root.

Menggunakan Tag Skrip

SDK siaran Web didistribusikan sebagai JavaScript perpustakaan dan dapat diambil di [https://web-broadcast.live-video.net/1.33.0/amazon-ivs-web-broadcast](https://web-broadcast.live-video.net/1.33.0/amazon-ivs-web-broadcast.js) .js.

Kelas dan enum yang didefinisikan dalam contoh di bawah ini dapat ditemukan pada objek `IVSBroadcastClient` global:

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

Menggunakan npm

Untuk menginstal npm paket:

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

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

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

Dukungan Rendering Sisi Server

Pustaka Tahapan SDK Siaran Web tidak dapat dimuat dalam konteks sisi server, karena mereferensikan primitif browser yang diperlukan untuk fungsi pustaka saat dimuat. Untuk mengatasinya, muat pustaka secara dinamis, seperti yang ditunjukkan dalam [Demo Siaran Web menggunakan Next dan React](#).

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 & Berlangganan dengan SDK Siaran Web IVS | Streaming Waktu Nyata

Dokumen ini membawa Anda melalui langkah-langkah yang terlibat dalam penerbitan dan berlangganan ke panggung menggunakan SDK siaran Web streaming real-time IVS.

Konsep

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

Stage

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 untuk melihat satu sama lain; misalnya, aplikasi obrolan video.

Implementasi yang lebih maju juga dimungkinkan. Misalnya, asumsikan aplikasi menyediakan `role` atribut saat membuat token dengan `CreateParticipantToken`. Aplikasi dapat menggunakan `attributes` properti `StageParticipantInfo` untuk berlangganan peserta secara selektif berdasarkan atribut yang disediakan server:

```
const strategy = {
```

```
shouldSubscribeToParticipant(participant) {
  switch (participant.attributes.role) {
    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.

Konfigurasi untuk Berlangganan Peserta

```
subscribeConfiguration(participant: StageParticipantInfo): SubscribeConfiguration
```

Jika peserta jarak jauh sedang berlangganan (lihat [Berlangganan Peserta](#)), SDK akan menanyakan aplikasi host tentang konfigurasi langganan khusus untuk peserta tersebut. Konfigurasi ini bersifat opsional dan memungkinkan aplikasi host untuk mengontrol aspek-aspek tertentu dari perilaku pelanggan. Untuk informasi tentang apa yang dapat dikonfigurasi, lihat [SubscribeConfiguration](#) di dokumentasi referensi SDK.

Berikut adalah contoh implementasi:

```
const strategy = {
  subscribeConfiguration: (participant) => {
    return {
      jitterBuffer: {
        minDelay: JitterBufferMinDelay.MEDIUM
      }
    }
  }
  // ... other strategy functions
}
```

Implementasi ini memperbarui penundaan minimum jitter-buffer untuk semua peserta berlangganan ke preset. MEDIUM

Seperti halnya `shouldSubscribeToParticipant`, implementasi yang lebih maju dimungkinkan. Yang diberikan `ParticipantInfo` dapat digunakan untuk memperbarui konfigurasi berlangganan secara selektif untuk peserta tertentu.

Sebaiknya gunakan perilaku default. Tentukan konfigurasi khusus hanya jika ada perilaku tertentu yang ingin Anda ubah.

Publikasi

```
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 seen/heard. (They also can use `publish/unpublish`, but that is much slower. `Mute/unmute` lebih disukai untuk kasus penggunaan di mana mengubah 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 untuk UI aplikasi host biasanya dapat didukung sepenuhnya oleh peristiwa. 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_ADAPTION_CHANGED, (participant, stream, isAdapting)
  => ())
stage.on(StageEvents.STAGE_STREAM_LAYERS_CHANGED, (participant, stream, layers) => ())
```

```
stage.on(StageEvents.STAGE_STREAM_LAYER_SELECTED, (participant, stream, layer, reason)
=> ())
stage.on(StageEvents.STAGE_STREAM_MUTE_CHANGED, (participant, stream) => {})
stage.on(StageEvents.STAGE_STREAM_SEI_MESSAGE_RECEIVED, (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 pembuatan token tambahan untuk panggung, khususnya untuk menerbitkan media screenshare. Gunakan `getDisplayMedia` dan batasi resolusi hingga maksimum 720p. Setelah itu, langkah-langkahnya mirip dengan menerbitkan kamera ke panggung.

```
// 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) => {
  let 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 dengan itu:

```
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

`requestQualityStats()` Metode ini menyediakan akses ke statistik WebRTC terperinci untuk aliran lokal dan jarak jauh. Ini tersedia pada keduanya `LocalStageStream` dan `RemoteStageStream` objek. Ini mengembalikan metrik kualitas yang komprehensif termasuk kualitas jaringan, statistik paket, informasi bitrate, dan metrik terkait bingkai.

Ini adalah metode asinkron yang dengannya Anda dapat mengambil statistik baik melalui `await` atau dengan merantai janji. Ini kembali `undefined` ketika statistik tidak tersedia; misalnya, aliran tidak aktif atau statistik internal tidak tersedia. Jika statistik tersedia, dan tergantung pada aliran (jarak jauh atau lokal, video atau audio), metode mengembalikan [LocalVideoStats](#), [LocalAudioStats](#), [RemoteVideoStats](#), atau [RemoteAudioStats](#) objek.

Perhatikan bahwa untuk streaming video dengan simulcast, array berisi beberapa objek stat (satu per lapisan).

Praktik Terbaik

- Frekuensi polling - Panggilan `requestQualityStats()` pada interval yang wajar (1-5 detik) untuk menghindari dampak kinerja

- Penanganan kesalahan - Selalu periksa apakah nilai yang dikembalikan undefined sebelum diproses
- Manajemen memori — Hapus intervals/timeouts saat aliran tidak lagi diperlukan
- Kualitas jaringan — Gunakan `networkQuality` untuk umpan balik pengguna mengenai kemungkinan degradasi yang disebabkan oleh jaringan. Lihat perinciannya di [NetworkQuality](#).

Contoh Penggunaan

```
// For local streams
const localStats = await localVideoStream.requestQualityStats();
const audioStats = await localAudioStream.requestQualityStats();

// For remote streams
const remoteVideoStats = await remoteVideoStream.requestQualityStats();
const remoteAudioStats = await remoteAudioStream.requestQualityStats();

// Example: Monitor stats every 10 seconds
const statsInterval = setInterval(async () => {
  const stats = await localVideoStream.requestQualityStats();
  if (stats) {
    // Note: If simulcast is enabled, you may receive multiple
    // stats records for each layer
    stats.forEach(layer => {
      const rid = layer.rid || 'default';
      console.log(`Layer ${rid}:`, {
        active: layer.active,
        networkQuality: layer.networkQuality,
        packetsSent: layer.packetsSent,
        bytesSent: layer.bytesSent,
        resolution: `${layer.frameWidth}x${layer.frameHeight}`,
        fps: layer.framesPerSecond
      });
    });
  }
}, 10000);
```

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.
 - Hal ini memungkinkan server untuk memilih rendisi mana yang akan dikirim ke peserta lain, berdasarkan keterbatasan jaringan mereka.
 - Ketika `simulcast` ditentukan bersama dengan `maxBitrate` and/or `maxFramerate` nilai, 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` operasi, Anda dapat melihat atribut di `StageParticipantInfo` properti:

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

Informasi Peningkatan Tambahan (SEI)

Unit Supplemental Enhancement Information (SEI) NAL digunakan untuk menyimpan metadata yang selaras dengan bingkai di samping video. Ini dapat digunakan saat menerbitkan dan berlangganan aliran video H.264. Payload SEI tidak dijamin sampai ke pelanggan, terutama dalam kondisi jaringan yang buruk. Karena muatan SEI menyimpan data secara langsung dalam struktur bingkai H.264, kemampuan ini tidak dapat dimanfaatkan untuk aliran audio saja.

Memasukkan Muatan SEI

Klien penerbitan dapat memasukkan muatan SEI ke aliran panggung yang sedang dipublikasikan dengan mengonfigurasi video mereka `LocalStageStream` untuk mengaktifkan `inBandMessaging` dan kemudian menjalankan metode `insertSeiMessage`. Perhatikan bahwa mengaktifkan `inBandMessaging` meningkatkan penggunaan memori SDK.

Muatan harus dari [ArrayBuffer](#) jenisnya. Ukuran muatan harus lebih besar dari 0KB dan kurang dari 1KB. Jumlah pesan SEI yang dimasukkan per detik tidak boleh melebihi 10KB per detik.

```
const config = {
  inBandMessaging: { enabled: true }
};
const vidStream = new LocalStageStream(videoTrack, config);
```

```
const payload = new TextEncoder().encode('hello world').buffer;
vidStream.insertSeiMessage(payload);
```

Mengulangi Muatan SEI

Opsional menyediakan `repeatCount` untuk mengulangi penyisipan muatan SEI untuk N frame berikutnya yang dikirim. Ini dapat membantu untuk mengurangi kerugian bawaan yang mungkin terjadi karena protokol transportasi UDP yang mendasari yang digunakan untuk mengirim video. Perhatikan nilai ini harus antara 0 dan 30. Klien yang menerima harus memiliki logika untuk menghapus duplikat pesan.

```
vidStream.insertSeiMessage(payload, { repeatCount: 5 }); // Optional config,
repeatCount must be between 0 and 30
```

Membaca Muatan SEI

Klien berlangganan dapat membaca muatan SEI dari penerbit yang menerbitkan video H.264 jika ada dengan mengonfigurasi pelanggan `SubscribeConfiguration` untuk mengaktifkan `inBandMessaging` dan mendengarkan acara, seperti yang ditunjukkan pada `StageEvents.STAGE_STREAM_SEI_MESSAGE_RECEIVED` contoh berikut:

```
const strategy = {
  subscribeConfiguration: (participant) => {
    return {
      inBandMessaging: {
        enabled: true
      }
    }
  }
  // ... other strategy functions
}

stage.on(StageEvents.STAGE_STREAM_SEI_MESSAGE_RECEIVED, (participant, seiMessage) => {
  console.log(seiMessage.payload, seiMessage.uuid);
});
```

Pengkodean Berlapis dengan Simulcast

Layered encoding dengan simulcast adalah fitur streaming real-time IVS yang memungkinkan penerbit mengirim beberapa lapisan video berkualitas berbeda, dan pelanggan untuk mengubah

lapisan tersebut secara dinamis atau manual. Fitur ini dijelaskan lebih lanjut dalam dokumen [Pengoptimalan Streaming](#).

Mengkonfigurasi Layered Encoding (Publisher)

Sebagai penerbit, untuk mengaktifkan pengkodean berlapis dengan simulcast, tambahkan konfigurasi berikut ke instantiasi saat Anda: `LocalStageStream`

```
// Enable Simulcast
let cameraStream = new LocalStageStream(cameraDevice, {
  simulcast: { enabled: true }
})
```

Bergantung pada resolusi input perangkat kamera Anda, sejumlah lapisan akan dikodekan dan dikirim seperti yang didefinisikan di bagian [Default Layers, Quality, dan Framerates](#) dari Pengoptimalan Streaming.

Selain itu, Anda dapat secara opsional mengonfigurasi lapisan individual dari dalam konfigurasi simulcast:

```
import { SimulcastLayerPresets } from 'amazon-ivs-web-broadcast'

// Enable Simulcast
let cameraStream = new LocalStageStream(cameraDevice, {
  simulcast: {
    enabled: true,
    layers: [
      SimulcastLayerPresets.DEFAULT_720,
      SimulcastLayerPresets.DEFAULT_360,
      SimulcastLayerPresets.DEFAULT_180,
    ]
  }
})
```

Sebagai alternatif, Anda dapat membuat konfigurasi lapisan kustom Anda sendiri hingga tiga lapisan. Jika Anda memberikan array kosong atau tidak ada nilai, default yang dijelaskan di atas digunakan. Lapisan dijelaskan dengan properti yang diperlukan berikut:

- `height`: number;
- `width`: number;
- `maxBitrateKbps`: number;

- `maxFramerate`: number;

Mulai dari preset, Anda dapat mengganti properti individual atau membuat konfigurasi yang sama sekali baru:

```
import { SimulcastLayerPresets } from 'amazon-ivs-web-broadcast'

const custom720pLayer = {
  ...SimulcastLayerPresets.DEFAULT_720,
  maxFramerate: 15,
}

const custom360pLayer = {
  maxBitrateKbps: 600,
  maxFramerate: 15,
  width: 640,
  height: 360,
}

// Enable Simulcast
let cameraStream = new LocalStageStream(cameraDevice, {
  simulcast: {
    enabled: true,
    layers: [
      custom720pLayer,
      custom360pLayer,
    ]
  }
})
```

Untuk nilai maksimum, batas, dan kesalahan yang dapat dipicu saat mengonfigurasi lapisan individual, lihat dokumentasi referensi SDK.

Mengonfigurasi Layered Encoding (Subscriber)

Sebagai pelanggan, tidak ada yang diperlukan untuk mengaktifkan pengkodean berlapis. Jika penerbit mengirim lapisan simulcast, maka secara default server secara dinamis beradaptasi di antara lapisan untuk memilih kualitas optimal berdasarkan perangkat pelanggan dan kondisi jaringan.

Atau, untuk memilih lapisan eksplisit yang dikirimkan penerbit, ada beberapa opsi, yang dijelaskan di bawah ini.

Opsi 1: Preferensi Kualitas Lapisan Awal

Dengan menggunakan `subscribeConfiguration` strategi, dimungkinkan untuk memilih lapisan awal apa yang ingin Anda terima sebagai pelanggan:

```
const strategy = {
  subscribeConfiguration: (participant) => {
    return {
      simulcast: {
        initialLayerPreference: InitialLayerPreference.LOWEST_QUALITY
      }
    }
  }
  // ... other strategy functions
}
```

Secara default, pelanggan selalu dikirim lapisan kualitas terendah terlebih dahulu; ini perlahan naik ke lapisan kualitas tertinggi. Ini mengoptimalkan konsumsi bandwidth pengguna akhir dan memberikan waktu terbaik untuk video, mengurangi pembekuan video awal bagi pengguna di jaringan yang lebih lemah.

Opsi ini tersedia untuk `InitialLayerPreference`:

- **LOWEST_QUALITY**— Server memberikan lapisan video dengan kualitas terendah terlebih dahulu. Ini mengoptimalkan konsumsi bandwidth, serta waktu ke media. Kualitas didefinisikan sebagai kombinasi ukuran, bitrate, dan framerate video. Misalnya, video 720p memiliki kualitas lebih rendah dari video 1080p.
- **HIGHEST_QUALITY**— Server memberikan lapisan video kualitas tertinggi terlebih dahulu. Ini mengoptimalkan kualitas tetapi dapat meningkatkan waktu ke media. Kualitas didefinisikan sebagai kombinasi ukuran, bitrate, dan framerate video. Misalnya, video 1080p memiliki kualitas lebih tinggi dari video 720p.

Catatan: Agar preferensi lapisan awal (`initialLayerPreference` panggilan) diterapkan, berlangganan ulang diperlukan karena pembaruan ini tidak berlaku untuk langganan aktif.

Opsi 2: Lapisan Pilihan untuk Stream

Setelah streaming dimulai, Anda dapat menggunakan metode `preferredLayerForStream` strategi. Metode strategi ini mengekspos peserta dan informasi aliran.

Metode strategi dapat dikembalikan dengan yang berikut:

- Objek layer secara langsung, berdasarkan apa yang `RemoteStageStream.getLayers` kembali
- String label objek lapisan, berdasarkan `StageStreamLayer.label`
- Tidak terdefinisi atau null, yang menunjukkan bahwa tidak ada lapisan yang harus dipilih, dan adaptasi dinamis lebih disukai

Misalnya, strategi berikut akan selalu membuat pengguna memilih lapisan video dengan kualitas terendah yang tersedia:

```
const strategy = {
  preferredLayerForStream: (participant, stream) => {
    return stream.getLowestQualityLayer();
  }
  // ... other strategy functions
}
```

Untuk mengatur ulang pemilihan lapisan dan kembali ke adaptasi dinamis, kembalikan null atau undefined dalam strategi. Dalam contoh ini `appState` adalah variabel dummy yang mewakili status aplikasi yang mungkin.

```
const strategy = {
  preferredLayerForStream: (participant, stream) => {
    if (appState.isAutoMode) {
      return null;
    } else {
      return appState.layerChoice
    }
  }
  // ... other strategy functions
}
```

Opsi 3: Pembantu RemoteStageStream Lapisan

`RemoteStageStream` memiliki beberapa pembantu yang dapat digunakan untuk membuat keputusan tentang pemilihan lapisan dan menampilkan pilihan yang sesuai untuk pengguna akhir:

- Layer Events — Di samping `ituStageEvents`, `RemoteStageStream` objek itu sendiri memiliki peristiwa yang mengkomunikasikan perubahan adaptasi layer dan simulcast:

- `stream.on(RemoteStageStreamEvents.ADAPTION_CHANGED, (isAdapting) => {})`
- `stream.on(RemoteStageStreamEvents.LAYERS_CHANGED, (layers) => {})`
- `stream.on(RemoteStageStreamEvents.LAYER_SELECTED, (layer, reason) => {})`
- Metode Layer — `RemoteStageStream` memiliki beberapa metode pembantu yang dapat digunakan untuk mendapatkan informasi tentang aliran dan lapisan yang disajikan. Metode ini tersedia di aliran jarak jauh yang disediakan dalam `preferredLayerForStream` strategi, serta aliran jarak jauh yang diekspos melalui `StageEvents.STAGE_PARTICIPANT_STREAMS_ADDED`.
 - `stream.getLayers`
 - `stream.getSelectedLayer`
 - `stream.getLowestQualityLayer`
 - `stream.getHighestQualityLayer`

Untuk detailnya, lihat `RemoteStageStream` kelas dalam [dokumentasi referensi SDK](#). Untuk `LAYER_SELECTED` alasannya, jika `UNAVAILABLE` dikembalikan, ini menunjukkan bahwa lapisan yang diminta tidak dapat dipilih. Pilihan upaya terbaik dilakukan sebagai gantinya, yang biasanya merupakan lapisan kualitas yang lebih rendah untuk menjaga stabilitas aliran.

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
      return;
    case StageConnectionState.CONNECTING:
      // handle establishing connection UI
      return;
    case StageConnectionState.CONNECTED:
      // SDK is connected to the Stage
  }
})
```

```

        return;
    case StageConnectionState.ERRORRED:
        // SDK encountered an error and lost its connection to the stage. Wait for
        CONNECTED.
        return;
    }
})

```

Secara umum, Anda dapat mengabaikan status kesalahan yang ditemui setelah berhasil bergabung dengan tahap, karena SDK akan mencoba memulihkan secara internal. Jika SDK melaporkan ERRORRED status dan tahapan tetap dalam CONNECTING status untuk jangka waktu yang lama (misalnya, 30 detik atau lebih), Anda mungkin terputus dari jaringan.

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 & Solusi yang Diketahui di SDK Siaran Web IVS | Streaming Waktu Nyata

Dokumen ini mencantumkan masalah yang diketahui yang mungkin Anda temui saat menggunakan SDK siaran Web streaming real-time Amazon IVS dan menyarankan solusi potensial.

- 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.

- Ketika `stage.leave()` dipanggil atau peserta jarak jauh pergi, kesalahan 404 DELETE muncul di konsol debug browser.

Solusi: Tidak ada. Ini adalah kesalahan yang tidak berbahaya.

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, OBS Virtual Camera 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 dari `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 di SDK Siaran Web IVS | Streaming Waktu Nyata

Bagian ini adalah ikhtisar kondisi kesalahan, bagaimana SDK siaran Web melaporkannya ke aplikasi, dan apa yang harus dilakukan aplikasi ketika kesalahan tersebut ditemui. Kesalahan dilaporkan oleh SDK kepada pendengar acara: `StageEvents.ERROR`

```
stage.on(StageEvents.ERROR, (error: StageError) => {  
  // log or handle errors here  
  console.log(`${error.code}, ${error.category}, ${error.message}`);  
});
```

Kesalahan Panggung

A `StageError` dilaporkan ketika SDK mengalami masalah yang tidak dapat dipulihkan dan umumnya memerlukan koneksi ulang and/or jaringan intervensi aplikasi untuk pulih.

Setiap dilaporkan `StageError` memiliki kode (atau `StageErrorCode`), pesan (string), dan kategori (`StageErrorCategory`). Masing-masing terkait dengan kategori operasi yang mendasarinya.

Kategori operasi kesalahan ditentukan berdasarkan apakah itu terkait dengan koneksi ke tahap (JOIN_ERROR), mengirim media ke tahap (PUBLISH_ERROR), atau menerima aliran media yang masuk dari tahap (SUBSCRIBE_ERROR).

Properti kode `StageError` melaporkan masalah spesifik:

Nama	Kode	Tindakan yang Direkomendasikan
TOKEN_MALFORMED	1	Buat token yang valid dan coba lagi membuat instance stage.
TOKEN_KEDALUWARSA	2	Buat token yang belum kedaluwarsa dan coba lagi membuat instance panggung.
BATAS WAKTU	3	Waktu operasi habis. Jika tahap ada dan token valid, kegagalan ini kemungkinan merupakan masalah jaringan. Dalam hal ini, tunggu konektivitas perangkat pulih.
FAILED	4	<p>Kondisi fatal ditemui ketika mencoba operasi. Periksa detail kesalahan.</p> <p>Jika tahap ada dan token valid, kegagalan ini kemungkinan merupakan masalah jaringan. Dalam hal ini, tunggu konektivitas perangkat pulih.</p> <p>Untuk sebagian besar kegagalan yang terkait dengan stabilitas jaringan, SDK akan mencoba lagi secara internal untuk jangka waktu hingga 30 detik sebelum memancarkan kesalahan GAGAL.</p>
MEMBATALKAN	5	Periksa kode aplikasi dan pastikan tidak ada pemanggilan berulang <code>joinrefreshStrategy</code> , atau <code>replaceStrategy</code> pemanggilan, yang dapat menyebabkan operasi berulang dimulai dan dibatalkan sebelum selesai.
STAGE_AT_CAPACITY	6	Kesalahan ini menunjukkan bahwa panggung atau akun Anda dalam kapasitas. Jika tahap telah

Nama	Kode	Tindakan yang Direkomendasikan
		mencapai batas pesertanya, coba operasi lagi ketika tahap tidak lagi dalam kapasitas, dengan menyegarkan strategi. Jika akun Anda telah mencapai langganan bersamaan atau kuota penayang bersamaan, kurangi penggunaan atau minta peningkatan kuota melalui konsol AWS Service Quotas.
CODEC_MISMATCH	7	Codec tidak didukung oleh panggung. Periksa browser dan platform untuk dukungan codec. Untuk streaming real-time IVS, browser harus mendukung codec H.264 untuk video dan codec Opus untuk audio.
TOKEN_NOT_ALLOWED	8	Token tidak memiliki izin untuk operasi. Buat ulang token dengan izin yang benar dan coba lagi.
STAGE_DELETED	9	Tidak ada; mencoba bergabung dengan tahap yang dihapus memicu kesalahan ini.
PARTISIPANT_TERPUT US	10	Tidak ada; mencoba bergabung dengan token peserta yang terputus memicu kesalahan ini.

StageError Contoh Penanganan

Gunakan StageError kode untuk menentukan apakah kesalahan disebabkan oleh token yang kedaluwarsa:

```
stage.on(StageEvents.ERROR, (error: StageError) => {
  if (error.code === StageError.TOKEN_EXPIRED) {
    // recreate the token and stage instance and re-join
  }
});
```

Kesalahan Jaringan saat Sudah Bergabung

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

Jika Anda menerbitkan and/or langganan, 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.

Negara Tersalah

Kami menyarankan Anda menggunakan status ini untuk pencatatan aplikasi dan untuk menampilkan pesan kepada pengguna yang memberi tahu mereka tentang masalah konektivitas ke panggung untuk peserta tertentu.

Publikasikan

SDK melaporkan `ERRORRED` saat publikasi gagal.

```
stage.on(StageEvents.STAGE_PARTICIPANT_PUBLISH_STATE_CHANGED, (participantInfo, state)
=> {
  if (state === StageParticipantPublishState.ERRORRED) {
    // Log and/or display message to user
  }
});
```

Langganan

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) {
    // Log and/or display message to user
  }
});
```

SDK Siaran IVS: Panduan Android | Streaming Waktu Nyata

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

Paket `com.amazonaws.ivs.broadcast` 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: [1.40.0 \(Catatan Rilis\)](#)

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/1.40.0/android/`.](#)

Contoh kode: [Lihat repositori contoh Android di GitHub: `https://github.com/aws-samples/amazon-ivs-real-time-streaming-android-samples`](#)

Persyaratan platform: Android 9.0+

Memulai dengan SDK Siaran Android IVS | Streaming Waktu Nyata

Dokumen ini membawa Anda melalui langkah-langkah yang terlibat dalam memulai dengan SDK siaran Android streaming real-time IVS.

Instal Perpustakaan

Ada beberapa cara untuk menambahkan library siaran Android Amazon IVS ke lingkungan pengembangan Android Anda: gunakan Gradle secara langsung, gunakan katalog versi Gradle, atau instal SDK secara manual.

Gunakan Gradle secara langsung: Tambahkan pustaka ke `build.gradle` file modul Anda, seperti yang ditunjukkan di sini (untuk versi terbaru SDK siaran IVS):

```
repositories {
    mavenCentral()
}
```

```
dependencies {  
    implementation 'com.amazonaws:ivs-broadcast:1.40.0:stages@aar'  
}
```

Gunakan katalog versi Gradle: Pertama sertakan ini dalam file modul Anda: `build.gradle`

```
implementation(libs.ivs){  
    artifact {  
        classifier = "stages"  
        type = "aar"  
    }  
}
```

Kemudian sertakan yang berikut ini dalam `libs.version.toml` file (untuk versi terbaru SDK siaran IVS):

```
[versions]  
ivs="1.40.0"  
  
[libraries]  
ivs = {module = "com.amazonaws:ivs-broadcast", version.ref = "ivs"}
```

Instal 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.

Juga izinkan kontrol SDK atas speakerphone: Terlepas dari metode penginstalan yang Anda pilih, tambahkan juga izin berikut ke manifes Anda, untuk memungkinkan SDK mengaktifkan dan menonaktifkan speakerphone:

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

Menggunakan SDK dengan Simbol Debug

Kami juga menerbitkan versi SDK siaran Android yang menyertakan simbol debug. Anda dapat menggunakan versi ini untuk meningkatkan kualitas laporan debug (jejak tumpukan) di Firebase Crashlytics, jika Anda mengalami crash di SDK broadcast IVS; yaitu, `libbroadcastcore.so`. Saat

Anda melaporkan crash ini ke tim IVS SDK, jejak tumpukan berkualitas lebih tinggi memudahkan untuk memperbaiki masalah.

Untuk menggunakan versi SDK ini, letakkan yang berikut ini di file build Gradle Anda:

```
implementation "com.amazonaws:ivs-broadcast:$version:stages-unstripped@aar"
```

Gunakan baris di atas alih-alih ini:

```
implementation "com.amazonaws:ivs-broadcast:$version:stages@aar"
```

Mengunggah Simbol ke Firebase Crashlytics

Pastikan file build Gradle Anda disiapkan untuk Firebase Crashlytics. Ikuti instruksi Google di sini:

<https://firebase.google.com/docs/crashlytics/ndk-laporan>

Pastikan untuk memasukkan `com.google.firebase:firebase-crashlytics-ndk` sebagai ketergantungan.

Saat membuat aplikasi Anda untuk rilis, plugin Firebase Crashlytics akan mengunggah simbol secara otomatis. Untuk mengunggah simbol secara manual, jalankan salah satu dari berikut ini:

```
gradle uploadCrashlyticsSymbolFileRelease
```

```
./gradlew uploadCrashlyticsSymbolFileRelease
```

(Tidak ada salahnya jika simbol diunggah dua kali, baik secara otomatis maupun manual.)

Mencegah Rilis Anda .apk dari Menjadi Lebih Besar

Sebelum mengemas `.apk` file rilis, Plugin Android Gradle secara otomatis mencoba menghapus informasi debug dari pustaka bersama (termasuk library SDK siaran IVS). `libbroadcastcore.so` Namun, terkadang hal ini tidak terjadi. Akibatnya, `.apk` file Anda bisa menjadi lebih besar dan Anda bisa mendapatkan pesan peringatan dari Plugin Android Gradle bahwa file tersebut tidak dapat menghapus simbol debug dan mengemas `.so` file apa adanya. Jika ini terjadi, lakukan hal berikut:

- Instal Android NDK. Versi terbaru apa pun akan berfungsi.

- Tambahkan `ndkVersion <your_installed_ndk_version_number>` ke `build.gradle` file aplikasi Anda. Lakukan ini bahkan jika aplikasi Anda sendiri tidak mengandung kode asli.

Untuk informasi selengkapnya, lihat [laporan masalah](#) ini.

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();
    }
}
```

```
}
```

Menerbitkan & Berlangganan dengan SDK Siaran Android IVS | Streaming Waktu Nyata

Dokumen ini membawa Anda melalui langkah-langkah yang terlibat dalam penerbitan dan berlangganan ke panggung menggunakan SDK siaran Android streaming real-time IVS.

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.

Stage

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 StageRenderer 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.

Konfigurasi untuk Berlangganan Peserta

```
SubscribeConfiguration subscribeConfigurationForParticipant(@NonNull Stage stage,  
    @NonNull ParticipantInfo participantInfo);
```

Jika peserta jarak jauh sedang berlangganan (lihat [Berlangganan Peserta](#)), SDK akan menanyakan aplikasi host tentang konfigurasi langganan khusus untuk peserta tersebut. Konfigurasi ini bersifat opsional dan memungkinkan aplikasi host untuk mengontrol aspek-aspek tertentu dari perilaku pelanggan. Untuk informasi tentang apa yang dapat dikonfigurasi, lihat [SubscribeConfiguration](#) di dokumentasi referensi SDK.

Berikut adalah contoh implementasi:

```
@Override  
public SubscribeConfiguration subscribeConfigurationForParticipant(@NonNull Stage stage,  
    @NonNull ParticipantInfo participantInfo) {  
    SubscribeConfiguration config = new SubscribeConfiguration();  
  
    config.jitterBuffer.setMinDelay(JitterBufferConfiguration.JitterBufferDelay.MEDIUM());  
  
    return config;  
}
```

Implementasi ini memperbarui penundaan minimum jitter-buffer untuk semua peserta berlangganan ke preset. MEDIUM

Seperti halnya `shouldSubscribeToParticipant`, implementasi yang lebih maju dimungkinkan. Yang diberikan `ParticipantInfo` dapat digunakan untuk memperbarui konfigurasi berlangganan secara selektif untuk peserta tertentu.

Sebaiknya gunakan perilaku default. Tentukan konfigurasi khusus hanya jika ada perilaku tertentu yang ingin Anda ubah.

Publikasi

```
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 seen/heard. (They also can use publish/unpublish, but that is much slower. Mute/unmute lebih disukai untuk kasus penggunaan di mana mengubah 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 mengetuk 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);

void onStreamAdaptionChanged(@NonNull Stage stage, @NonNull ParticipantInfo
    participantInfo, @NonNull RemoteStageStream stream, boolean adaption);

void onStreamLayersChanged(@NonNull Stage stage, @NonNull ParticipantInfo
    participantInfo, @NonNull RemoteStageStream stream, @NonNull
    List<RemoteStageStream.Layer> layers);

void onStreamLayerSelected(@NonNull Stage stage, @NonNull ParticipantInfo
    participantInfo, @NonNull RemoteStageStream stream, @Nullable RemoteStageStream.Layer
    layer, @NonNull RemoteStageStream.LayerSelectedReason reason);
```

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` ke `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` operasi, 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());
    }
}
}

```

Sematkan Pesan

`embedMessage` metode ini `ImageDevice` memungkinkan Anda untuk memasukkan muatan metadata langsung ke bingkai video selama penerbitan. Ini memungkinkan pesan yang disinkronkan bingkai

untuk aplikasi waktu nyata. Penyematan pesan hanya tersedia saat menggunakan SDK untuk penerbitan real-time (bukan penerbitan latensi rendah).

Pesan yang disematkan tidak dijamin sampai ke pelanggan karena mereka disematkan langsung dalam bingkai video dan ditransmisikan melalui UDP, yang tidak menjamin pengiriman paket. Kehilangan paket selama transmisi dapat mengakibatkan pesan hilang, terutama dalam kondisi jaringan yang buruk. Untuk mengurangi ini, `embedMessage` metode ini menyertakan `repeatCount` parameter yang menduplikasi pesan di beberapa frame berturut-turut, meningkatkan keandalan pengiriman. Kemampuan ini hanya tersedia untuk streaming video.

Menggunakan EmbedMessage

Klien penerbitan dapat menyematkan muatan pesan ke dalam aliran video mereka menggunakan `embedMessage` metode aktif. ImageDevice Ukuran muatan harus lebih besar dari 0KB dan kurang dari 1KB. Jumlah pesan tertanam yang dimasukkan per detik tidak boleh melebihi 10KB per detik.

```
val surfaceSource: SurfaceSource = imageStream.device as SurfaceSource
val message = "hello world"
val messageBytes = message.toByteArray(StandardCharsets.UTF_8)

try {
    surfaceSource.embedMessage(messageBytes, 0)
} catch (e: BroadcastException) {
    Log.e("EmbedMessage", "Failed to embed message: ${e.message}")
}
```

Mengulangi Muatan Pesan

Gunakan `repeatCount` untuk menduplikasi pesan di beberapa frame untuk meningkatkan keandalan. Nilai ini harus antara 0 dan 30. Klien penerima harus memiliki logika untuk menghapus duplikat pesan.

```
try {
    surfaceSource.embedMessage(messageBytes, 5)
    // repeatCount: 0-30, receiving clients should handle duplicates
} catch (e: BroadcastException) {
    Log.e("EmbedMessage", "Failed to embed message: ${e.message}")
}
```

Membaca Pesan Tertanam

Lihat “Dapatkan Informasi Peningkatan Tambahan (SEI)” di bawah ini untuk mengetahui cara membaca pesan yang disematkan dari aliran masuk.

Dapatkan Informasi Peningkatan Tambahan (SEI)

Unit Supplemental Enhancement Information (SEI) NAL digunakan untuk menyimpan metadata yang selaras dengan bingkai di samping video. Klien berlangganan dapat membaca muatan SEI dari penerbit yang menerbitkan video H.264 dengan memeriksa `embeddedMessages` properti pada objek yang keluar dari penerbit. `ImageDeviceFrame` `ImageDevice` Untuk melakukan ini, dapatkan `publisherImageDevice`, lalu amati setiap frame melalui callback yang disediakan `setOnFrameCallback`, seperti yang ditunjukkan pada contoh berikut:

```
// in a StageRenderer's onStreamsAdded function, after acquiring the new ImageStream
val imageDevice = imageStream.device as ImageDevice
imageDevice.setOnFrameCallback(object : ImageDevice.FrameCallback {
    override fun onFrame(frame: ImageDeviceFrame) {
        for (message in frame.embeddedMessages) {
            if (message is UserDataUnregisteredSeiMessage) {
                val seiMessageBytes = message.data
                val seiMessageUUID = message.uuid

                // interpret the message's data based on the UUID
            }
        }
    }
})
```

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();
}

```

Pengkodean Berlapis dengan Simulcast

Layered encoding dengan simulcast adalah fitur streaming real-time IVS yang memungkinkan penerbit mengirim beberapa lapisan video berkualitas berbeda, dan pelanggan untuk mengonfigurasi lapisan tersebut secara dinamis atau manual. Fitur ini dijelaskan lebih lanjut dalam dokumen [Pengoptimalan Streaming](#).

Mengkonfigurasi Layered Encoding (Publisher)

Sebagai penerbit, untuk mengaktifkan pengkodean berlapis dengan simulcast, tambahkan konfigurasi berikut ke instantiasi saat Anda: `LocalStageStream`

```

// Enable Simulcast
StageVideoConfiguration config = new StageVideoConfiguration();
config.simulcast.setEnabled(true);

ImageLocalStageStream cameraStream = new ImageLocalStageStream(frontCamera, config);

// Other Stage implementation code

```

Bergantung pada resolusi yang Anda tetapkan pada konfigurasi video, sejumlah lapisan akan dikodekan dan dikirim seperti yang didefinisikan di bagian [Default Layers, Quality, dan Framerates](#) dari Pengoptimalan Streaming.

Selain itu, Anda dapat secara opsional mengonfigurasi lapisan individual dari dalam konfigurasi simulcast:

```
// Enable Simulcast
StageVideoConfiguration config = new StageVideoConfiguration();
config.simulcast.setEnabled(true);

List<StageVideoConfiguration.Simulcast.Layer> simulcastLayers = new ArrayList<>();
simulcastLayers.add(StagePresets.SimulcastLocalLayer.DEFAULT_720);
simulcastLayers.add(StagePresets.SimulcastLocalLayer.DEFAULT_180);

config.simulcast.setLayers(simulcastLayers);

ImageLocalStageStream cameraStream = new ImageLocalStageStream(frontCamera, config);

// Other Stage implementation code
```

Sebagai alternatif, Anda dapat membuat konfigurasi lapisan kustom Anda sendiri hingga tiga lapisan. Jika Anda memberikan array kosong atau tidak ada nilai, default yang dijelaskan di atas digunakan. Lapisan dijelaskan dengan setter properti wajib berikut:

- `setSize: Vec2;`
- `setMaxBitrate: integer;`
- `setMinBitrate: integer;`
- `setTargetFramerate: integer;`

Mulai dari preset, Anda dapat mengganti properti individual atau membuat konfigurasi yang sama sekali baru:

```
// Enable Simulcast
StageVideoConfiguration config = new StageVideoConfiguration();
config.simulcast.setEnabled(true);

List<StageVideoConfiguration.Simulcast.Layer> simulcastLayers = new ArrayList<>();
```

```
// Configure high quality layer with custom framerate
StageVideoConfiguration.Simulcast.Layer customHiLayer =
    StagePresets.SimulcastLocalLayer.DEFAULT_720;
customHiLayer.setTargetFramerate(15);

// Add layers to the list
simulcastLayers.add(customHiLayer);
simulcastLayers.add(StagePresets.SimulcastLocalLayer.DEFAULT_180);

config.simulcast.setLayers(simulcastLayers);

ImageLocalStageStream cameraStream = new ImageLocalStageStream(frontCamera, config);

// Other Stage implementation code
```

Untuk nilai maksimum, batas, dan kesalahan yang dapat dipicu saat mengonfigurasi lapisan individual, lihat dokumentasi referensi SDK.

Mengonfigurasi Layered Encoding (Subscriber)

Sebagai pelanggan, tidak ada yang diperlukan untuk mengaktifkan pengkodean berlapis. Jika penerbit mengirim lapisan simulcast, maka secara default server secara dinamis beradaptasi di antara lapisan untuk memilih kualitas optimal berdasarkan perangkat pelanggan dan kondisi jaringan.

Atau, untuk memilih lapisan eksplisit yang dikirimkan penerbit, ada beberapa opsi, yang dijelaskan di bawah ini.

Opsi 1: Preferensi Kualitas Lapisan Awal

Dengan menggunakan `subscribeConfigurationForParticipant` strategi, dimungkinkan untuk memilih lapisan awal apa yang ingin Anda terima sebagai pelanggan:

```
@Override
public SubscribeConfiguration subscribeConfigurationForParticipant(@NonNull Stage stage,
    @NonNull ParticipantInfo participantInfo) {
    SubscribeConfiguration config = new SubscribeConfiguration();

    config.simulcast.setInitialLayerPreference(SubscribeSimulcastConfiguration.InitialLayerPreference);

    return config;
}
```

Secara default, pelanggan selalu dikirim lapisan kualitas terendah terlebih dahulu; ini perlahan naik ke lapisan kualitas tertinggi. Ini mengoptimalkan konsumsi bandwidth pengguna akhir dan memberikan waktu terbaik untuk video, mengurangi pembekuan video awal bagi pengguna di jaringan yang lebih lemah.

Opsi ini tersedia untuk `InitialLayerPreference`:

- `LOWEST_QUALITY`— Server memberikan lapisan video dengan kualitas terendah terlebih dahulu. Ini mengoptimalkan konsumsi bandwidth, serta waktu ke media. Kualitas didefinisikan sebagai kombinasi ukuran, bitrate, dan framerate video. Misalnya, video 720p memiliki kualitas lebih rendah dari video 1080p.
- `HIGHEST_QUALITY`— Server memberikan lapisan video kualitas tertinggi terlebih dahulu. Ini mengoptimalkan kualitas tetapi dapat meningkatkan waktu ke media. Kualitas didefinisikan sebagai kombinasi ukuran, bitrate, dan framerate video. Misalnya, video 1080p berkualitas lebih tinggi dari video 720p.

Catatan: Agar preferensi lapisan awal (`setInitialLayerPreference` panggilan) diterapkan, berlangganan ulang diperlukan karena pembaruan ini tidak berlaku untuk langganan aktif.

Opsi 2: Lapisan Pilihan untuk Stream

Metode `preferredLayerForStream` strategi memungkinkan Anda memilih lapisan setelah aliran dimulai. Metode strategi ini menerima peserta dan informasi aliran, sehingga Anda dapat memilih lapisan participant-by-participant berdasarkan. SDK memanggil metode ini sebagai respons terhadap peristiwa tertentu, seperti ketika lapisan aliran berubah, status peserta berubah, atau aplikasi host menyegarkan strategi.

Metode strategi mengembalikan `RemoteStageStream.Layer` objek, yang dapat menjadi salah satu dari berikut:

- Objek layer, seperti yang dikembalikan oleh `RemoteStageStream.getLayers`.
- `null`, yang menunjukkan bahwa tidak ada lapisan yang harus dipilih dan adaptasi dinamis lebih disukai.

Misalnya, strategi berikut akan selalu membuat pengguna memilih lapisan video dengan kualitas terendah yang tersedia:

```
@Nullable
```

```
@Override
public RemoteStageStream.Layer preferredLayerForStream(@NonNull Stage stage, @NonNull
ParticipantInfo participantInfo, @NonNull RemoteStageStream stream) {
    return stream.getLowestQualityLayer();
}
```

Untuk mengatur ulang pemilihan lapisan dan kembali ke adaptasi dinamis, kembalikan null atau undefined dalam strategi. Dalam contoh ini, `appState` adalah variabel placeholder yang mewakili status aplikasi host.

```
@Nullable
@Override
public RemoteStageStream.Layer preferredLayerForStream(@NonNull Stage stage, @NonNull
ParticipantInfo participantInfo, @NonNull RemoteStageStream stream) {
    if (appState.isAutoMode) {
        return null;
    } else {
        return appState.layerChoice;
    }
}
```

Opsi 3: Pembantu RemoteStageStream Lapisan

`RemoteStageStream` memiliki beberapa pembantu yang dapat digunakan untuk membuat keputusan tentang pemilihan lapisan dan menampilkan pilihan yang sesuai untuk pengguna akhir:

- **Layer Events** — Di samping `ituStageRenderer`, `RemoteStageStream.Listener` memiliki peristiwa yang mengkomunikasikan perubahan adaptasi layer dan simulcast:
 - `void onAdaptionChanged(boolean adaption)`
 - `void onLayersChanged(@NonNull List<Layer> layers)`
 - `void onLayerSelected(@Nullable Layer layer, @NonNull LayerSelectedReason reason)`
- **Metode Layer** — `RemoteStageStream` memiliki beberapa metode pembantu yang dapat digunakan untuk mendapatkan informasi tentang aliran dan lapisan yang disajikan. Metode ini tersedia di aliran jarak jauh yang disediakan dalam `preferredLayerForStream` strategi, serta aliran jarak jauh yang diekspos melalui `StageRenderer.onStreamsAdded`.
 - `stream.getLayers`
 - `stream.getSelectedLayer`

- `stream.getLowestQualityLayer`
- `stream.getHighestQualityLayer`
- `stream.getLayersWithConstraints`

Untuk detailnya, lihat `RemoteStageStream` kelas dalam [dokumentasi referensi SDK](#). Untuk `LayerSelected` alasannya, jika `UNAVAILABLE` dikembalikan, ini menunjukkan bahwa lapisan yang diminta tidak dapat dipilih. Pilihan upaya terbaik dilakukan sebagai gantinya, yang biasanya merupakan lapisan kualitas yang lebih rendah untuk menjaga stabilitas aliran.

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 telah 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 & Solusi yang Diketahui di SDK Siaran Android IVS | Streaming Waktu Nyata

Dokumen ini mencantumkan masalah yang diketahui yang mungkin Anda temui saat menggunakan SDK siaran Android streaming real-time Amazon IVS dan menyarankan solusi potensial.

- 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 di SDK Siaran Android IVS | Streaming Waktu Nyata

Bagian ini adalah ikhtisar kondisi kesalahan, bagaimana SDK siaran Android streaming real-time IVS melaporkannya ke aplikasi, dan apa yang harus dilakukan aplikasi ketika kesalahan tersebut ditemui.

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 publish/subscribe status dan ditangani oleh SDK, yang mencoba ulang operasi. publish/subscribe

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 `stage.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, `publish/subscribe` negara 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. `publish/subscribe` Selama percobaan ulang, `publish/subscribe` negara 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.

IVS Broadcast SDK: 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: [1.40.0 \(Catatan Rilis\)](#)

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/1.40.0/ios/>.](#)

Contoh kode: [Lihat contoh repositori iOS di GitHub: <https://github.com/aws-samples/amazon-ivs-real-time-streaming-ios-samples>](https://github.com/aws-samples/amazon-ivs-real-time-streaming-ios-samples)

Persyaratan platform: iOS 14+

Memulai dengan SDK Siaran iOS IVS | Streaming Waktu Nyata

Dokumen ini membawa Anda melalui langkah-langkah yang terlibat dalam memulai dengan SDK siaran iOS streaming real-time IVS.

Instal Perpustakaan

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

Direkomendasikan: Integrasikan Broadcast SDK (Swift Package Manager)

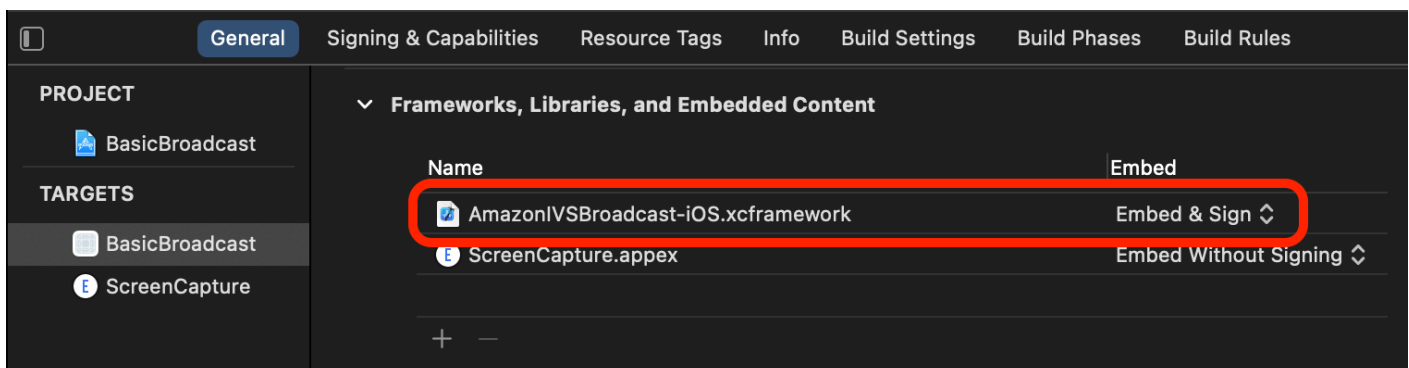
1. Unduh file Package.swift dari <https://broadcast.live-video.net/1.40.0/Package.swift>.

2. Dalam proyek Anda, buat direktori baru bernama Amazon IVSBroadcast dan tambahkan ke kontrol versi.
3. Tempatkan file Package.swift yang diunduh di direktori baru.
4. Di Xcode, buka File > Add Package Dependencies dan pilih Add Local...
5. Arahkan ke dan pilih IVSBroadcast direktori Amazon yang Anda buat, lalu pilih Add Package.
6. Saat diminta untuk Memilih Produk Paket untuk Amazon IVSBroadcast, pilih IVSBroadcastTahapan Amazon sebagai Produk Package Anda dengan menetapkan target aplikasi Anda di bagian Tambahkan ke Target.
7. Pilih Add Package.

Penting: SDK siaran streaming real-time IVS mencakup semua fitur SDK siaran streaming latensi rendah IVS. Tidak mungkin untuk mengintegrasikan keduanya SDKs dalam proyek yang sama.

Pendekatan Alternatif: Instal Kerangka Secara Manual

1. Unduh versi terbaru dari <https://broadcast.live-video.net/1.40.0/AmazonIVSBroadcast-Stages.xcframework.zip>.
2. Ekstrak konten 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
}
```

Menerbitkan & Berlangganan dengan SDK Siaran iOS IVS | Streaming Waktu Nyata

Dokumen ini membawa Anda melalui langkah-langkah yang terlibat dalam penerbitan dan berlangganan ke panggung menggunakan SDK siaran iOS streaming real-time IVS.

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.

Stage

`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.

Konfigurasi untuk Berlangganan Peserta

```
func stage(_ stage: IVSStage, subscribeConfigurationForParticipant participant:
  IVSParticipantInfo) -> IVSSubscribeConfiguration
```

Jika peserta jarak jauh sedang berlangganan (lihat [Berlangganan Peserta](#)), SDK akan menanyakan aplikasi host tentang konfigurasi langganan khusus untuk peserta tersebut. Konfigurasi ini bersifat

opsional dan memungkinkan aplikasi host untuk mengontrol aspek-aspek tertentu dari perilaku pelanggan. Untuk informasi tentang apa yang dapat dikonfigurasi, lihat [SubscribeConfiguration](#) di dokumentasi referensi SDK.

Berikut adalah contoh implementasi:

```
func stage(_ stage: IVSStage, subscribeConfigurationForParticipant participant:
  IVSParticipantInfo) -> IVSSubscribeConfiguration {
    let config = IVSSubscribeConfiguration()

    try! config.jitterBuffer.setMinDelay(.medium())

    return config
}
```

Implementasi ini memperbarui penundaan minimum jitter-buffer untuk semua peserta berlangganan ke preset. MEDIUM

Seperti halnya `shouldSubscribeToParticipant`, implementasi yang lebih maju dimungkinkan. Yang diberikan `ParticipantInfo` dapat digunakan untuk memperbarui konfigurasi berlangganan secara selektif untuk peserta tertentu.

Sebaiknya gunakan perilaku default. Tentukan konfigurasi khusus hanya jika ada perilaku tertentu yang ingin Anda ubah.

Publikasi

```
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 seen/heard. (They also can use publish/unpublish, but that is much slower. Mute/unmute lebih disukai untuk kasus penggunaan di mana mengubah 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

IVSStageRendererProtokol 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?)

func stage(_ stage: IVSStage, participant: IVSParticipantInfo, stream:
  IVSRemoteStageStream, didChangeStreamAdaption adaption: Bool)

func stage(_ stage: IVSStage, participant: IVSParticipantInfo, stream:
  IVSRemoteStageStream, didChange layers: [IVSRemoteStageStreamLayer])

func stage(_ stage: IVSStage, participant: IVSParticipantInfo, stream:
  IVSRemoteStageStream, didSelect layer: IVSRemoteStageStreamLayer?, reason:
  IVSRemoteStageStream.LayerSelectedReason)
```

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 and create a stream
let microphone = devices.compactMap({ $0 as? IVSMicrophone }).first!
let microphoneStream = IVSLocalStageStream(device: microphone)

// Configure the audio manager to use the videoChat preset, which is optimized for bi-directional communication, including echo cancellation.
IVSStageAudioManager.sharedInstance().setPreset(.videoChat)

// 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` operasi, 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)")
}
}
```

Sematkan Pesan

`embedMessage` metode pada `IVSImage Perangkat` memungkinkan Anda memasukkan muatan metadata langsung ke bingkai video selama penerbitan. Ini memungkinkan pesan yang disinkronkan bingkai untuk aplikasi waktu nyata. Penyematkan pesan hanya tersedia saat menggunakan SDK untuk penerbitan real-time (bukan penerbitan latensi rendah).

Pesan yang disematkan tidak dijamin sampai ke pelanggan karena mereka disematkan langsung dalam bingkai video dan ditransmisikan melalui UDP, yang tidak menjamin pengiriman paket. Kehilangan paket selama transmisi dapat mengakibatkan pesan hilang, terutama dalam kondisi jaringan yang buruk. Untuk mengurangi ini, `embedMessage` metode ini menyertakan `repeatCount` parameter yang menduplikasi pesan di beberapa frame berturut-turut, meningkatkan keandalan pengiriman. Kemampuan ini hanya tersedia untuk streaming video.

Menggunakan `EmbedMessage`

Klien penerbitan dapat menyematkan muatan pesan ke aliran video mereka menggunakan `embedMessage` metode di `IVSImage Perangkat`. Ukuran muatan harus lebih besar dari 0KB dan kurang dari 1KB. Jumlah pesan tertanam yang dimasukkan per detik tidak boleh melebihi 10KB per detik.

```
let imageDevice: IVSImageDevice = imageStream.device as! IVSImageDevice
let messageData = Data("hello world".utf8)

do {
    try imageDevice.embedMessage(messageData, withRepeatCount: 0)
} catch {
    print("Failed to embed message: \$(error)")
}
```

Mengulangi Muatan Pesan

Gunakan `repeatCount` untuk menduplikasi pesan di beberapa frame untuk meningkatkan keandalan. Nilai ini harus antara 0 dan 30. Klien yang menerima harus memiliki logika untuk menghapus duplikat pesan.

```
try imageDevice.embedMessage(messageData, withRepeatCount: 5)

// repeatCount: 0-30, receiving clients should handle duplicates
```

Membaca Pesan Tertanam

Lihat “Dapatkan Informasi Peningkatan Tambahan (SEI)” di bawah ini untuk cara membaca pesan yang disematkan dari aliran masuk.

Dapatkan Informasi Peningkatan Tambahan (SEI)

Unit Supplemental Enhancement Information (SEI) NAL digunakan untuk menyimpan metadata yang selaras dengan bingkai di samping video. Klien berlangganan dapat membaca muatan SEI dari penerbit yang menerbitkan video H.264 dengan memeriksa `embeddedMessages` properti pada objek yang keluar dari penerbit. `IVSImageDeviceFrame` `IVSImageDevice` Untuk melakukan ini, dapatkan `publisherIVSImageDevice`, lalu amati setiap frame melalui callback yang disediakan `setOnFrameCallback`, seperti yang ditunjukkan pada contoh berikut:

```
// in an IVSStageRenderer's stage:participant:didAddStreams: function, after acquiring
// the new IVSImageStream

let imageDevice: IVSImageDevice? = imageStream.device as? IVSImageDevice
imageDevice?.setOnFrameCallback { frame in
    for message in frame.embeddedMessages {
        if let seiMessage = message as? IVSUserDataUnregisteredSEIMessage {
            let seiMessageData = seiMessage.data
            let seiMessageUUID = seiMessage.UUID

            // interpret the message's data based on the UUID
        }
    }
}
```

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 `IVSStrategy` 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()`.

Pengkodean Berlapis dengan Simulcast

Layered encoding dengan simulcast adalah fitur streaming real-time IVS yang memungkinkan penerbit mengirim beberapa lapisan video berkualitas berbeda, dan pelanggan untuk mengonfigurasi lapisan tersebut secara dinamis atau manual. Fitur ini dijelaskan lebih lanjut dalam dokumen [Pengoptimalan Streaming](#).

Mengkonfigurasi Layered Encoding (Publisher)

Sebagai penerbit, untuk mengaktifkan pengkodean berlapis dengan simulcast, tambahkan konfigurasi berikut ke instantiasi saat Anda: `IVSLocalStageStream`

```
// Enable Simulcast
let config = IVSLocalStageStreamVideoConfiguration()
config.simulcast.enabled = true

let cameraStream = IVSLocalStageStream(device: camera, configuration: config)

// Other Stage implementation code
```

Bergantung pada resolusi yang Anda tetapkan pada konfigurasi video, sejumlah lapisan akan dikodekan dan dikirim seperti yang didefinisikan di bagian [Lapisan Default, Kualitas, dan Framerates](#) dari Pengoptimalan Streaming.

Selain itu, Anda dapat secara opsional mengonfigurasi lapisan individual dari dalam konfigurasi simulcast:

```
// Enable Simulcast
let config = IVSLocalStageStreamVideoConfiguration()
config.simulcast.enabled = true
```

```
let layers = [
    IVSStagePresets.simulcastLocalLayer().default720(),
    IVSStagePresets.simulcastLocalLayer().default180()
]

try config.simulcast.setLayers(layers)

let cameraStream = IVSLocalStageStream(device: camera, configuration: config)

// Other Stage implementation code
```

Sebagai alternatif, Anda dapat membuat konfigurasi lapisan kustom Anda sendiri hingga tiga lapisan. Jika Anda memberikan array kosong atau tidak ada nilai, default yang dijelaskan di atas digunakan. Lapisan dijelaskan dengan setter properti wajib berikut:

- `setSize: CGSize;`
- `setMaxBitrate: integer;`
- `setMinBitrate: integer;`
- `setTargetFramerate: float;`

Mulai dari preset, Anda dapat mengganti properti individual atau membuat konfigurasi yang sama sekali baru:

```
// Enable Simulcast
let config = IVSLocalStageStreamVideoConfiguration()
config.simulcast.enabled = true

let customHiLayer = IVSStagePresets.simulcastLocalLayer().default720()
try customHiLayer.setTargetFramerate(15)

let layers = [
    customHiLayer,
    IVSStagePresets.simulcastLocalLayer().default180()
]

try config.simulcast.setLayers(layers)

let cameraStream = IVSLocalStageStream(device: camera, configuration: config)

// Other Stage implementation code
```

Untuk nilai maksimum, batas, dan kesalahan yang dapat dipicu saat mengonfigurasi lapisan individual, lihat dokumentasi referensi SDK.

Mengonfigurasi Layered Encoding (Subscriber)

Sebagai pelanggan, tidak ada yang diperlukan untuk mengaktifkan pengkodean berlapis. Jika penerbit mengirim lapisan simulcast, maka secara default server secara dinamis beradaptasi antara lapisan untuk memilih kualitas optimal berdasarkan perangkat pelanggan dan kondisi jaringan.

Atau, untuk memilih lapisan eksplisit yang dikirimkan penerbit, ada beberapa opsi, yang dijelaskan di bawah ini.

Opsi 1: Preferensi Kualitas Lapisan Awal

Dengan menggunakan `subscribeConfigurationForParticipant` strategi, dimungkinkan untuk memilih lapisan awal apa yang ingin Anda terima sebagai pelanggan:

```
func stage(_ stage: IVSStage, subscribeConfigurationForParticipant participant:
  IVSParticipantInfo) -> IVSSubscribeConfiguration {
    let config = IVSSubscribeConfiguration()

    config.simulcast.initialLayerPreference = .lowestQuality

    return config
}
```

Secara default, pelanggan selalu dikirim lapisan kualitas terendah terlebih dahulu; ini perlahan naik ke lapisan kualitas tertinggi. Ini mengoptimalkan konsumsi bandwidth pengguna akhir dan memberikan waktu terbaik untuk video, mengurangi pembekuan video awal bagi pengguna di jaringan yang lebih lemah.

Opsi ini tersedia untuk `InitialLayerPreference`:

- `lowestQuality`— Server memberikan lapisan video dengan kualitas terendah terlebih dahulu. Ini mengoptimalkan konsumsi bandwidth, serta waktu ke media. Kualitas didefinisikan sebagai kombinasi ukuran, bitrate, dan framerate video. Misalnya, video 720p memiliki kualitas lebih rendah dari video 1080p.
- `highestQuality`— Server memberikan lapisan video kualitas tertinggi terlebih dahulu. Ini mengoptimalkan kualitas tetapi dapat meningkatkan waktu ke media. Kualitas didefinisikan sebagai kombinasi ukuran, bitrate, dan framerate video. Misalnya, video 1080p berkualitas lebih tinggi dari video 720p.

Catatan: Agar preferensi lapisan awal (`initialLayerPreference`) diterapkan, berlangganan ulang diperlukan karena pembaruan ini tidak berlaku untuk langganan aktif.

Opsi 2: Lapisan Pilihan untuk Stream

Metode `preferredLayerForStream` strategi memungkinkan Anda memilih lapisan setelah aliran dimulai. Metode strategi ini menerima peserta dan informasi aliran, sehingga Anda dapat memilih lapisan `participant-by-participant` berdasarkan. SDK memanggil metode ini sebagai respons terhadap peristiwa tertentu, seperti ketika lapisan aliran berubah, status peserta berubah, atau aplikasi host menyegarkan strategi.

Metode strategi mengembalikan `IVSRemoteStageStreamLayer` objek, yang dapat menjadi salah satu dari berikut:

- Objek layer, seperti yang dikembalikan oleh `IVSRemoteStageStream.layers`.
- `null`, yang menunjukkan bahwa tidak ada lapisan yang harus dipilih dan adaptasi dinamis lebih disukai.

Misalnya, strategi berikut akan selalu membuat pengguna memilih lapisan video berkualitas terendah yang tersedia:

```
func stage(_ stage: IVSStage, participant: IVSParticipantInfo, preferredLayerFor
stream: IVSRemoteStageStream) -> IVSRemoteStageStreamLayer? {
    return stream.lowestQualityLayer
}
```

Untuk mengatur ulang pemilihan lapisan dan kembali ke adaptasi dinamis, kembalikan `null` atau `undefined` dalam strategi. Dalam contoh ini, `appState` adalah variabel placeholder yang mewakili status aplikasi host.

```
func stage(_ stage: IVSStage, participant: IVSParticipantInfo, preferredLayerFor
stream: IVSRemoteStageStream) -> IVSRemoteStageStreamLayer? {
    If appState.isAutoMode {
        return nil
    } else {
        return appState.layerChoice
    }
}
```

Opsi 3: Pembantu RemoteStageStream Lapisan

IVSRemoteStageStream memiliki beberapa pembantu yang dapat digunakan untuk membuat keputusan tentang pemilihan lapisan dan menampilkan pilihan yang sesuai untuk pengguna akhir:

- **Layer Events** — Di samping itu `IVSStageRenderer`, `IVSRemoteStageStreamDelegate` memiliki peristiwa yang mengkomunikasikan perubahan adaptasi layer dan simulcast:
 - `func stream(_ stream: IVSRemoteStageStream, didChangeAdaption adaption: Bool)`
 - `func stream(_ stream: IVSRemoteStageStream, didChange layers: [IVSRemoteStageStreamLayer])`
 - `func stream(_ stream: IVSRemoteStageStream, didSelect layer: IVSRemoteStageStreamLayer?, reason: IVSRemoteStageStream.LayerSelectedReason)`
- **Metode Layer** — `IVSRemoteStageStream` memiliki beberapa metode pembantu yang dapat digunakan untuk mendapatkan informasi tentang aliran dan lapisan yang disajikan. Metode ini tersedia di aliran jarak jauh yang disediakan dalam `preferredLayerForStream` strategi, serta aliran jarak jauh yang diekspos melalui `func stage(_ stage: IVSStage, participant: IVSParticipantInfo, didAdd streams: [IVSStageStream])`.
 - `stream.layers`
 - `stream.selectedLayer`
 - `stream.lowestQualityLayer`
 - `stream.highestQualityLayer`
 - `stream.layers(with: IVSRemoteStageStreamLayerConstraints)`

Untuk detailnya, lihat `IVSRemoteStageStream` kelas dalam [dokumentasi referensi SDK](#). Untuk `LayerSelected` alasannya, jika `UNAVAILABLE` dikembalikan, ini menunjukkan bahwa lapisan yang diminta tidak dapat dipilih. Pilihan upaya terbaik dilakukan sebagai gantinya, yang biasanya merupakan lapisan kualitas yang lebih rendah untuk menjaga stabilitas aliran.

Siarkan Panggung ke Saluran IVS

Untuk menyiarkan panggung, buat yang terpisah `IVSBroadcastSession` dan kemudian ikuti instruksi biasa untuk penyiaran dengan SDK, 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 & Solusi yang Diketahui di SDK Siaran iOS IVS | Streaming Waktu Nyata

Dokumen ini mencantumkan masalah yang diketahui yang mungkin Anda temui saat menggunakan SDK siaran iOS streaming real-time Amazon IVS dan menyarankan solusi potensial.

- 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 di SDK Siaran iOS IVS | Streaming Waktu Nyata

Bagian ini adalah ikhtisar kondisi kesalahan, bagaimana IVS real-time streaming iOS broadcast SDK melaporkannya ke aplikasi, dan apa yang harus dilakukan aplikasi ketika kesalahan tersebut ditemui.

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 publish/subscribe status dan ditangani oleh SDK, yang mencoba ulang operasi. publish/subscribe

Anda dapat memeriksa properti ini:

```
let nsError = error as NSError
if nsError.userInfo[IVSBroadcastErrorIsFatalKey] as? Bool == true {
    // the error is fatal
}
```

Bergabung Error

Token Cacat

Ini terjadi ketika token panggung salah bentuk.

SDK melempar pengecualian Swift dengan kode kesalahan = 1000 dan `IVSBroadcastErrorIsFatalKey = 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 = YA. `IVSBroadcastErrorIsFatalKey`

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 `stage(didChange connectionState, withError error)` dengan kode kesalahan = 1026 dan `IVSBroadcast ErrorIsFatalKey = 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.

SDK memanggil `stage(didChange connectionState, withError error)` dengan kode kesalahan = 1300 dan `IVSBroadcast ErrorIsFatalKey = 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.

SDK memanggil `stage(didChange connectionState, withError error)` dengan kode kesalahan = 1300 dan `IVSBroadcast ErrorIsFatalKey nilai = 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, `publish/subscribe` negara adalah `ATTEMPTING_PUBLISH/ATTEMPTING_SUBSCRIBE`. Jika upaya coba lagi berhasil, status berubah menjadi `PUBLISHED/SUBSCRIBED`.

Panggilan SDK `IVSErrorDelegate:didEmitError` dengan kode kesalahan yang relevan dan `IVSBroadcastErrorIsFatalKey == 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. `publish/subscribe` Selama percobaan ulang, `publish/subscribe` negara adalah `ATTEMPTING_PUBLISH/ATTEMPTING_SUBSCRIBE`. Jika upaya coba lagi berhasil, status berubah menjadi `PUBLISHED/SUBSCRIBED`.

SDK memanggil `didEmitError` dengan kode kesalahan = 1400 dan `IVSBroadcast ErrorIsFatalKey = 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: Perangkat Campuran

Perangkat campuran adalah perangkat audio dan video yang mengambil beberapa sumber input dan menghasilkan satu output. Perangkat pencampuran adalah fitur canggih yang memungkinkan

Anda menentukan dan mengelola beberapa elemen (video) dan trek audio di layar. Anda dapat menggabungkan video dan audio dari berbagai sumber seperti kamera, mikrofon, tangkapan layar, serta audio dan video yang dihasilkan oleh aplikasi Anda. Anda dapat menggunakan transisi untuk memindahkan sumber-sumber ini di sekitar video yang Anda streaming ke IVS, dan menambah dan menghapus sumber di tengah aliran.

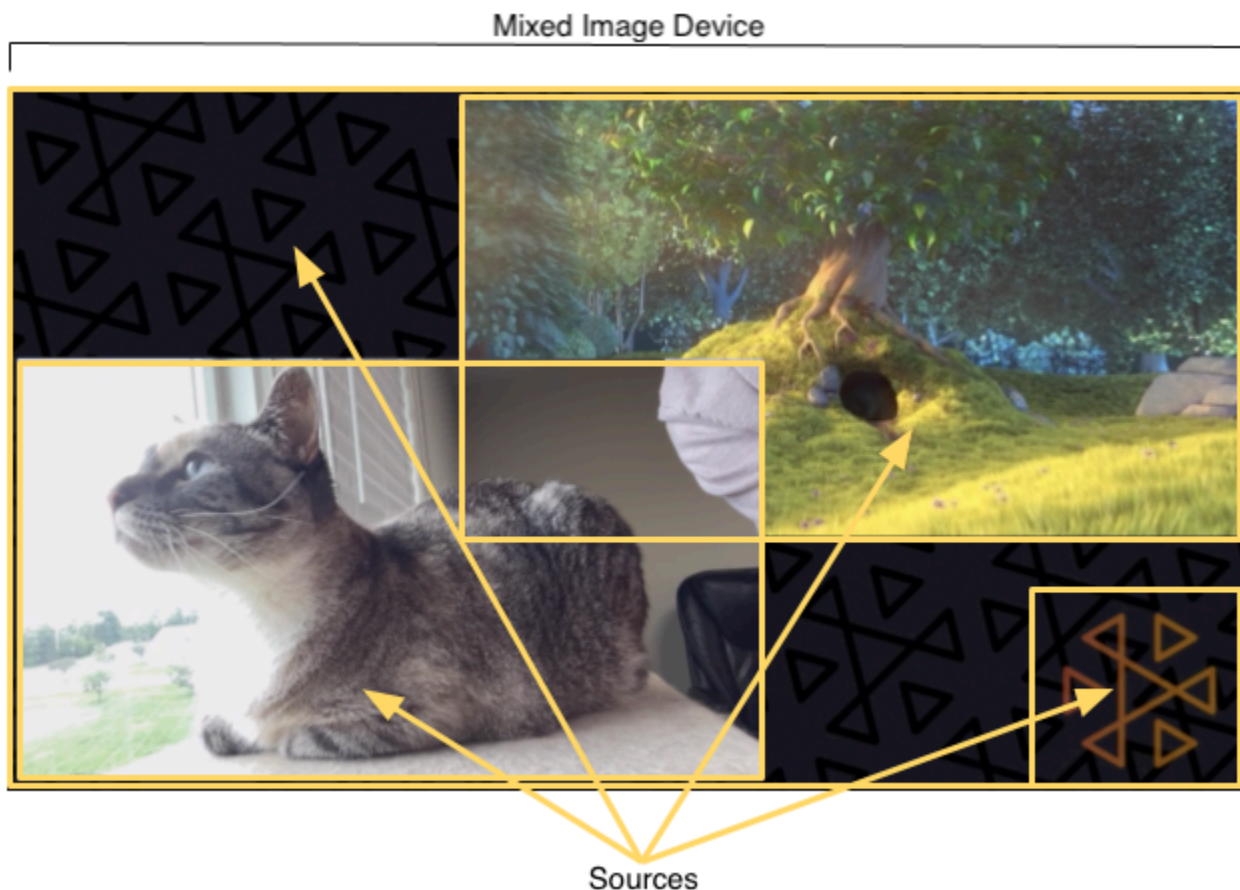
Perangkat campuran datang dalam rasa gambar dan audio. Untuk membuat perangkat gambar campuran, hubungi:

`DeviceDiscovery.createMixedImageDevice()` di Android

`IVSDeviceDiscovery.createMixedImageDevice()` di iOS

Perangkat yang dikembalikan dapat dilampirkan ke `BroadcastSession` (streaming latensi rendah) atau `Stage` (streaming waktu nyata), seperti perangkat lainnya.

Terminologi



Istilah	Deskripsi
Perangkat	Komponen perangkat keras atau perangkat lunak yang menghasilkan input audio atau gambar. Contoh perangkat adalah mikrofon, kamera, headset Bluetooth, dan perangkat virtual seperti tangkapan layar atau input gambar khusus.
Perangkat Campuran	<p>A Device yang dapat dilampirkan <code>BroadcastSession</code> seperti yang lain <code>Device</code>, tetapi dengan tambahan APIs yang memungkinkan <code>Source</code> objek ditambahkan. Perangkat campuran memiliki mixer internal yang menggabungkan audio atau gambar, menghasilkan audio output tunggal dan aliran gambar.</p> <p>Perangkat campuran datang dalam ragam audio atau gambar.</p>
Konfigurasi perangkat campuran	Objek konfigurasi untuk perangkat campuran. Untuk perangkat gambar campuran, ini mengonfigurasi properti seperti dimensi dan framerate. Untuk perangkat audio campuran, ini mengonfigurasi jumlah saluran.
Sumber	<p>Wadah yang mendefinisikan posisi elemen visual di layar dan properti trek audio dalam campuran audio. Perangkat campuran dapat dikonfigurasi dengan nol atau lebih sumber. Sumber diberikan konfigurasi yang mempengaruhi bagaimana media sumber digunakan. Gambar di atas menunjukkan empat sumber gambar:</p> <ul style="list-style-type: none"> • Kiri bawah dengan input kamera • Kanan atas dengan masukan film • Kanan bawah dengan logo Amazon IVS • Gambar latar belakang layar penuh
Konfigurasi Sumber	Objek konfigurasi untuk sumber yang masuk ke perangkat campuran. Objek konfigurasi lengkap dijelaskan di bawah ini..
Transisi	<p>Untuk memindahkan slot ke posisi baru atau mengubah beberapa propertinya, gunakan <code>MixedDevice.transitionToConfiguration()</code> . Metode ini membutuhkan:</p> <ul style="list-style-type: none"> • Konfigurasi sumber baru yang mewakili status berikutnya untuk sumber. • Durasi yang menentukan berapa lama animasi harus diambil, relatif terhadap timeline video. Jika durasi diatur ke 0, transisi terjadi pada frame berikutnya yang dicampur.

Istilah	Deskripsi
	<ul style="list-style-type: none"> • Callback opsional yang memberi tahu Anda saat animasi selesai. Callback mungkin berguna untuk merantai animasi.

Perangkat Audio Campuran

Konfigurasi

`MixedAudioDeviceConfiguration` di Android

`IVSMixedAudioDeviceConfiguration` di iOS

Nama	Tipe	Deskripsi
<code>channels</code>	Bilangan Bulat	Jumlah saluran output dari mixer audio. Nilai yang valid: 1, 2. 1 adalah audio mono; 2, audio stereo. Default: 2.

Konfigurasi Sumber

`MixedAudioDeviceSourceConfiguration` di Android

`IVSMixedAudioDeviceSourceConfiguration` di iOS

Nama	Tipe	Deskripsi
<code>gain</code>	Desimal	Keuntungan audio. Ini adalah pengganda, jadi nilai apa pun di atas 1 meningkatkan keuntungan; nilai apa pun di bawah 1, menurunkannya. Nilai yang valid: 0-2. Default: 1.

Perangkat Gambar Campuran

Konfigurasi

`MixedImageDeviceConfiguration` di Android

IVSMixedImageDeviceConfiguration di iOS

Nama	Tipe	Deskripsi
<code>size</code>	Vec2	Ukuran kanvas video.
<code>targetFramerate</code>	Bilangan Bulat	Jumlah frame target per detik untuk perangkat campuran. Rata-rata, nilai ini harus dipenuhi, tetapi sistem dapat menjatuhkan frame dalam keadaan tertentu (misalnya, beban CPU atau GPU yang tinggi).
<code>transparencyEnabled</code>	Boolean	Ini memungkinkan pencampuran menggunakan alpha properti pada konfigurasi sumber gambar. Mengatur ini untuk <code>true</code> meningkatkan konsumsi memori dan CPU. Default: <code>false</code> .

Konfigurasi Sumber

MixedImageDeviceSourceConfiguration di Android

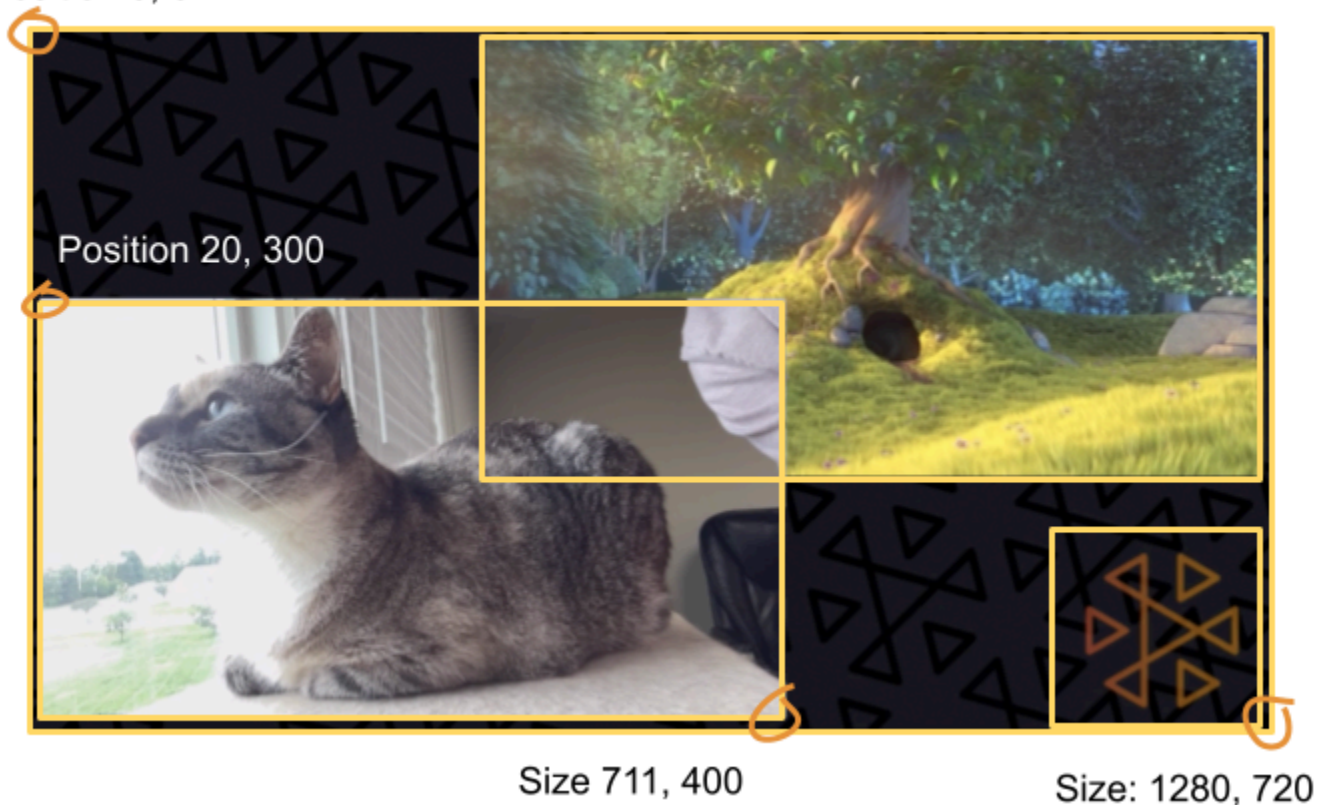
IVSMixedImageDeviceSourceConfiguration di iOS

Nama	Tipe	Deskripsi
<code>alpha</code>	Desimal	Alpha dari slot. Ini multiplikatif dengan nilai alfa apa pun pada gambar. Nilai yang valid: 0-1. 0 sepenuhnya transparan dan 1 sepenuhnya buram. Default: 1.
<code>aspect</code>	AspectMode	Mode rasio aspek untuk gambar apa pun yang dirender di slot. Nilai valid: <ul style="list-style-type: none"> <code>Fill</code>— Pertahankan rasio aspek gambar tetapi isi slotnya. Gambar dipotong jika diperlukan. <code>Fit</code>— Pertahankan rasio aspek gambar tetapi paskan seluruh gambar ke dalam slot. Slot mungkin memiliki kotak surat atau kotak pilar jika perlu. <code>letter/pillarbox</code> Ada di <code>fillColor</code> jika nilai itu ditetapkan; jika tidak, transparan (yang mungkin

Nama	Tipe	Deskripsi
		<p>tampak hitam jika warna kanvas di belakang gambar berwarna hitam).</p> <ul style="list-style-type: none"> • None— Jangan mempertahankan rasio aspek gambar. Gambar diskalakan agar sesuai dengan dimensi slot. <p>Default: <code>Fit</code></p>
<code>fillColor</code>	<code>Vec4</code>	Isi warna yang akan digunakan aspect <code>Fit</code> ketika rasio aspek slot dan gambar tidak cocok. Formatnya adalah (merah, hijau, biru, alfa). Nilai valid (untuk setiap saluran): 0-1. Default: (0, 0, 0, 0).
<code>position</code>	<code>Vec2</code>	Posisi slot (dalam piksel), relatif terhadap sudut kiri atas kanvas. Asal slot juga kiri atas.
<code>size</code>	<code>Vec2</code>	Ukuran slot, dalam piksel. Menyetel nilai ini juga disetel <code>matchCanvasSize</code> ke <code>false</code> . Default: (0, 0); Namun, karena <code>matchCanvasSize</code> defaultnya <code>true</code> , ukuran slot yang dirender adalah ukuran kanvas, bukan (0, 0).
<code>zIndex</code>	Desimal	Pemesanan slot relatif. Slot dengan <code>zIndex</code> nilai yang lebih tinggi digambar di atas slot dengan <code>zIndex</code> nilai yang lebih rendah.

Membuat dan Mengkonfigurasi Perangkat Gambar Campuran

Position 0, 0



Di sini, kami membuat adegan yang mirip dengan yang ada di awal panduan ini, dengan tiga elemen di layar:

- Slot kiri bawah untuk kamera.
- Slot kanan bawah untuk hamparan logo.
- Slot kanan atas untuk film.

Perhatikan bahwa asal kanvas adalah sudut kiri atas dan ini sama untuk slot. Oleh karena itu, memposisikan slot di (0, 0) menempatkannya di sudut kiri atas dengan seluruh slot terlihat.

iOS

```
let deviceDiscovery = IVSDeviceDiscovery()
let mixedImageConfig = IVSMixedImageDeviceConfiguration()
mixedImageConfig.size = CGSize(width: 1280, height: 720)
try mixedImageConfig.setTargetFramerate(60)
mixedImageConfig.isTransparencyEnabled = true
```

```
let mixedImageDevice = deviceDiscovery.createMixedImageDevice(with: mixedImageConfig)

// Bottom Left
let cameraConfig = IVSMixedImageDeviceSourceConfiguration()
cameraConfig.size = CGSize(width: 320, height: 180)
cameraConfig.position = CGPoint(x: 20, y: mixedImageConfig.size.height -
    cameraConfig.size.height - 20)
cameraConfig.zIndex = 2
let camera = deviceDiscovery.listLocalDevices().first(where: { $0 is IVSCamera }) as?
    IVSCamera
let cameraSource = IVSMixedImageDeviceSource(configuration: cameraConfig, device:
    camera)
mixedImageDevice.add(cameraSource)

// Top Right
let streamConfig = IVSMixedImageDeviceSourceConfiguration()
streamConfig.size = CGSize(width: 640, height: 320)
streamConfig.position = CGPoint(x: mixedImageConfig.size.width -
    streamConfig.size.width - 20, y: 20)
streamConfig.zIndex = 1
let streamDevice = deviceDiscovery.createImageSource(withName: "stream")
let streamSource = IVSMixedImageDeviceSource(configuration: streamConfig, device:
    streamDevice)
mixedImageDevice.add(streamSource)

// Bottom Right
let logoConfig = IVSMixedImageDeviceSourceConfiguration()
logoConfig.size = CGSize(width: 320, height: 180)
logoConfig.position = CGPoint(x: mixedImageConfig.size.width - logoConfig.size.width -
    20,
                                y: mixedImageConfig.size.height - logoConfig.size.height
    - 20)
logoConfig.zIndex = 3
let logoDevice = deviceDiscovery.createImageSource(withName: "logo")
let logoSource = IVSMixedImageDeviceSource(configuration: logoConfig, device:
    logoDevice)
mixedImageDevice.add(logoSource)
```

Android

```
val deviceDiscovery = DeviceDiscovery(this /* context */)
val mixedImageConfig = MixedImageDeviceConfiguration().apply {
    setSize(BroadcastConfiguration.Vec2(1280f, 720f))
}
```

```
        setTargetFramerate(60)
        setEnableTransparency(true)
    }
    val mixedImageDevice = deviceDiscovery.createMixedImageDevice(mixedImageConfig)

    // Bottom Left
    val cameraConfig = MixedImageDeviceSourceConfiguration().apply {
        setSize(BroadcastConfiguration.Vec2(320f, 180f))
        setPosition(BroadcastConfiguration.Vec2(20f, mixedImageConfig.size.y - size.y -
        20))
        setZIndex(2)
    }
    val camera = deviceDiscovery.listLocalDevices().firstNotNullOf { it as? CameraSource }
    val cameraSource = MixedImageDeviceSource(cameraConfig, camera)
    mixedImageDevice.addSource(cameraSource)

    // Top Right
    val streamConfig = MixedImageDeviceSourceConfiguration().apply {
        setSize(BroadcastConfiguration.Vec2(640f, 320f))
        setPosition(BroadcastConfiguration.Vec2(mixedImageConfig.size.x - size.x - 20,
        20f))
        setZIndex(1)
    }
    val streamDevice = deviceDiscovery.createImageInputSource(streamConfig.size)
    val streamSource = MixedImageDeviceSource(streamConfig, streamDevice)
    mixedImageDevice.addSource(streamSource)

    // Bottom Right
    val logoConfig = MixedImageDeviceSourceConfiguration().apply {
        setSize(BroadcastConfiguration.Vec2(320f, 180f))
        setPosition(BroadcastConfiguration.Vec2(mixedImageConfig.size.x - size.x - 20,
        mixedImageConfig.size.y - size.y - 20))
        setZIndex(1)
    }
    val logoDevice = deviceDiscovery.createImageInputSource(logoConfig.size)
    val logoSource = MixedImageDeviceSource(logoConfig, logoDevice)
    mixedImageDevice.addSource(logoSource)
```

Menghapus Sumber

Untuk menghapus sumber, panggil `MixedDevice.remove` dengan `Source` objek yang ingin Anda hapus.

Animasi dengan Transisi

Metode transisi menggantikan konfigurasi sumber dengan konfigurasi baru. Penggantian ini dapat dianimasikan dari waktu ke waktu dengan menetapkan durasi lebih tinggi dari 0, dalam hitungan detik.

Properti apa yang bisa dianimasikan?

Tidak semua properti dalam struktur slot dapat dianimasikan. Properti apa pun berdasarkan tipe Float dapat dianimasikan; properti lain berlaku baik di awal atau akhir animasi.

Nama	Bisakah itu dianimasikan?	Titik Dampak
<code>Audio.gain</code>	Ya	Diinterpolasi
<code>Image.alpha</code>	Ya	Diinterpolasi
<code>Image.aspect</code>	Tidak	Akhiri
<code>Image.fillColor</code>	Ya	Diinterpolasi
<code>Image.position</code>	Ya	Diinterpolasi
<code>Image.size</code>	Ya	Diinterpolasi
<code>Image.zIndex</code>	Ya	Tidak Diketahui

Catatan: Pesawat `zIndex` bergerak 2D melalui ruang 3D, sehingga transisi terjadi ketika dua bidang menyeberang di beberapa titik di tengah animasi. Ini bisa dihitung, tetapi itu tergantung pada `zIndex` nilai awal dan akhir. Untuk transisi yang lebih lancar, gabungkan ini dengan `alpha`.

Contoh Sederhana

Di bawah ini adalah contoh pengambilalihan kamera layar penuh menggunakan konfigurasi yang ditentukan di atas dalam [Membuat dan Mengkonfigurasi Perangkat Gambar Campuran](#). Ini dianimasikan lebih dari 0,5 detik.

iOS

```
// Continuing the example from above, modifying the existing cameraConfig object.
cameraConfig.size = CGSize(width: 1280, height: 720)
cameraConfig.position = CGPoint.zero
cameraSource.transition(to: cameraConfig, duration: 0.5) { completed in
    if completed {
        print("Animation completed")
    } else {
        print("Animation interrupted")
    }
}
```

Android

```
// Continuing the example from above, modifying the existing cameraConfig object.
cameraConfig.setSize(BroadcastConfiguration.Vec2(1280f, 720f))
cameraConfig.setPosition(BroadcastConfiguration.Vec2(0f, 0f))
cameraSource.transitionToConfiguration(cameraConfig, 500) { completed ->
    if (completed) {
        print("Animation completed")
    } else {
        print("Animation interrupted")
    }
}
```

Mencerminkan Siaran

Untuk mencerminkan perangkat gambar terlampir dalam siaran ke arah ini...	Gunakan nilai negatif untuk...
Horizontal	Lebar slot
Vertikal	Ketinggian slot

Untuk mencerminkan perangkat gambar terlampir dalam siaran ke arah ini...

Gunakan nilai negatif untuk...

Baik horisontal maupun vertikal

Baik lebar dan tinggi slot

Posisi perlu disesuaikan dengan nilai yang sama, untuk menempatkan slot pada posisi yang benar saat dicerminkan.

Di bawah ini adalah contoh untuk mencerminkan siaran secara horisontal dan vertikal.

iOS

Pencerminan horisontal:

```
let cameraSource = IVSMixedImageDeviceSourceConfiguration()
cameraSource.size = CGSize(width: -320, height: 720)
// Add 320 to position x since our width is -320
cameraSource.position = CGPoint(x: 320, y: 0)
```

Pencerminan vertikal:

```
let cameraSource = IVSMixedImageDeviceSourceConfiguration()
cameraSource.size = CGSize(width: 320, height: -720)
// Add 720 to position y since our height is -720
cameraSource.position = CGPoint(x: 0, y: 720)
```

Android

Pencerminan horisontal:

```
val cameraConfig = MixedImageDeviceSourceConfiguration().apply {
    setSize(BroadcastConfiguration.Vec2(-320f, 180f))
    // Add 320f to position x since our width is -320f
    setPosition(BroadcastConfiguration.Vec2(320f, 0f))
}
```

Pencerminan vertikal:

```
val cameraConfig = MixedImageDeviceSourceConfiguration().apply {
```

```
setSize(BroadcastConfiguration.Vec2(320f, -180f))
// Add 180f to position y since our height is -180f
setPosition(BroadcastConfiguration.Vec2(0f, 180f))
}
```

Catatan: Pencerminan ini berbeda dengan `setMirrored` metode di `ImagePreviewView` (Android) dan `IVSImagePreviewView` (iOS). Metode itu hanya memengaruhi tampilan pratinjau lokal di perangkat dan tidak memengaruhi siaran.

SDK Siaran IVS: Pertukaran Token | Streaming Waktu Nyata

Pertukaran token memungkinkan Anda untuk meningkatkan atau menurunkan kemampuan token peserta dan memperbarui atribut token dalam SDK siaran seluler, tanpa mengharuskan peserta untuk terhubung kembali. Ini berguna untuk skenario seperti co-hosting, di mana peserta dapat memulai dengan kemampuan berlangganan saja dan kemudian membutuhkan kemampuan publikasi.

Pembatasan:

- Pertukaran token hanya berfungsi dengan token yang dibuat di server Anda menggunakan [key pair](#). Ini tidak berfungsi dengan token yang dibuat melalui [CreateParticipantToken API](#).
- Jika Anda menggunakan pertukaran token untuk mengubah atribut yang mendorong server-side-composition tata letak (seperti `featuredParticipantAttribute` dan `participantOrderAttribute`), tata letak komposisi aktif tidak akan diperbarui hingga peserta tersambung kembali.

Bertukar Token

Pertukaran token sangat mudah: panggil `exchangeToken` API pada `IVSStage` objek `Stage/` dan berikan token baru. Jika token baru berbeda dari token sebelumnya, kemampuan token baru segera dievaluasi. `capabilities` Misalnya, jika token sebelumnya tidak memiliki `publish` kemampuan dan token baru melakukannya, fungsi strategi panggung untuk penerbitan dipanggil, memungkinkan aplikasi host untuk memutuskan apakah mereka ingin segera mempublikasikan dengan kemampuan baru, atau menunggu. Hal yang sama berlaku untuk kemampuan yang dihapus: jika token sebelumnya memiliki `publish` kemampuan dan token baru tidak, peserta segera membatalkan publikasi tanpa menggunakan fungsi strategi panggung untuk penerbitan.

Saat bertukar token, token sebelumnya dan baru harus memiliki nilai yang sama untuk bidang `payload` berikut:

- `topic`
- `resource`
- `jti`
- `whip_url`
- `events_url`

Bidang ini tidak dapat diubah. Bertukar token yang memodifikasi bidang yang tidak dapat diubah menghasilkan SDK segera menolak pertukaran.

Bidang yang tersisa dapat diubah, termasuk:

- `attributes`
- `capabilities`
- `user`
- `_id`
- `iat`
- `exp`

iOS

```
let stage = try IVSStage(token: originalToken, strategy: self)
stage.join()
stage.exchangeToken(newToken)
```

Android

```
val stage = Stage(context, originalToken, strategy)
stage.join()
stage.exchangeToken(newToken)
```

Menerima Pembaruan

Fungsi `diStageRenderer/IVSStageRenderer` menerima pembaruan tentang peserta jarak jauh yang sudah dipublikasikan yang bertukar token mereka untuk memperbarui `userId`.

`attributes` Peserta jarak jauh yang belum menerbitkan akan diperbarui `userId` dan `attributes` diekspos melalui fungsi `ParticipantJoined/participantDidJoinrenderer` yang ada jika mereka akhirnya mempublikasikannya.

iOS

```
class MyStageRenderer: NSObject, IVSStageRenderer {
    func stage(_ stage: IVSStage, participantMetadataDidUpdate participant:
    IVSParticipantInfo) {
        // participant will be a new IVSParticipantInfo instance with updated
        properties.
    }
}
```

Android

```
private val stageRenderer = object : StageRenderer {
    override fun onParticipantMetadataUpdated(stage: Stage, participantInfo:
    ParticipantInfo) {
        // participantInfo will be a new ParticipantInfo instance with updated
        properties.
    }
}
```

Visibilitas Pembaruan

Saat peserta menukar token untuk memperbarui `userId` atau `attributes`, visibilitas perubahan ini bergantung pada status penerbitan mereka saat ini:

- Jika peserta tidak menerbitkan: Pembaruan diproses secara diam-diam. Jika mereka akhirnya mempublikasikan, semua SDKs akan menerima yang diperbarui `userId` dan `attributes` sebagai bagian dari acara publikasi awal.
- Jika peserta sudah menerbitkan: Pembaruan disiarkan segera. Namun, hanya ponsel SDKs v1.37.0+ yang menerima notifikasi. Peserta di SDK web, seluler yang lebih lama SDKs, dan komposisi sisi server tidak melihat perubahan sampai peserta tidak menerbitkan dan menerbitkan kembali.

Tabel ini menjelaskan matriks dukungan:

Negara Peserta	Pengamat: SDK Seluler 1.37.0+	Pengamat: Seluler Lama SDKs, SDK Web, Komposisi Sisi Server
Tidak menerbitkan (lalu dimulai)	# Terlihat (saat dipublikasikan melalui acara yang bergabung dengan peserta)	# Terlihat (saat dipublikasikan melalui acara yang bergabung dengan peserta)
Sudah menerbitkan (tidak pernah menerbitkan kembali)	# Terlihat (segera melalui acara yang diperbarui metadata peserta)	# Tidak Terlihat
Sudah menerbitkan (tidak menerbitkan dan menerbitkan kembali)	# Terlihat (segera melalui acara yang diperbarui metadata peserta)	## Akhirnya Terlihat (saat dipublikasikan ulang melalui acara yang bergabung dengan peserta)

SDK Siaran IVS: Sumber Gambar Kustom | Streaming Waktu Nyata

Sumber input gambar khusus memungkinkan aplikasi untuk menyediakan input gambarnya sendiri ke SDK siaran, alih-alih terbatas pada kamera preset. Sumber gambar khusus dapat sesederhana tanda air semi-transparan atau adegan “segera kembali” statis, atau memungkinkan aplikasi untuk melakukan pemrosesan khusus tambahan seperti menambahkan filter kecantikan ke kamera.

Saat 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 harus mengelola kamera.

Android

Setelah Anda membuat `DeviceDiscovery` sesi, buat sumber input gambar:

```
CustomImageSource imageSource = deviceDiscovery.createImageInputSource(new
BroadcastConfiguration.Vec2(1280, 720));
```

Metode ini mengembalikan `CustomImageSource`, yang merupakan sumber gambar yang didukung oleh Android [Surface](#) standar. Subclass `SurfaceSource` dapat diubah ukurannya dan diputar. Anda juga dapat membuat `ImagePreviewView` untuk menampilkan pratinjau isinya.

Untuk mengambil yang mendasarinya `Surface`:

```
Surface surface = surfaceSource.getInputSurface();
```

Ini `Surface` dapat digunakan sebagai buffer output untuk produsen gambar seperti `Camera2`, `OpenGL ES`, dan perpustakaan lainnya. Kasus penggunaan paling sederhana adalah langsung menggambar bitmap statis atau warna ke dalam kanvas `Surface`. Namun, banyak pustaka (seperti pustaka filter kecantikan) menyediakan metode yang memungkinkan aplikasi menentukan eksternal untuk rendering. `Surface` Anda dapat menggunakan metode seperti itu untuk meneruskan ini `Surface` ke pustaka filter, yang memungkinkan pustaka mengeluarkan bingkai yang diproses agar sesi siaran dapat dialirkan.

Ini `CustomImageSource` dapat dibungkus dalam `LocalStageStream` dan dikembalikan oleh `StageStrategy` untuk mempublikasikan ke `aStage`.

iOS

Setelah Anda membuat `DeviceDiscovery` sesi, buat sumber input gambar:

```
let customSource = broadcastSession.createImageSource(withName: "customSourceName")
```

Metode ini mengembalikan `IVSCustomImageSource`, yang merupakan sumber gambar yang memungkinkan aplikasi untuk mengirimkan `CMSampleBuffers` 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 kustom akan dialirkan ke Panggung:

```
customSource.onSampleBuffer(sampleBuffer)
```

Untuk streaming video, gunakan metode ini dalam panggilan balik. 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.

`IVSCustomImageSource` dapat dibungkus dalam `IVSLocalStageStream` dan dikembalikan oleh `IVSStageStrategy` untuk mempublikasikan ke `aStage`.

SDK Siaran IVS: Sumber Audio Kustom | Streaming Waktu Nyata

Catatan: Panduan ini hanya berlaku untuk SDK siaran Android streaming real-time IVS. Informasi untuk iOS dan web SDKs akan dipublikasikan di masa depan.

Sumber input audio khusus memungkinkan aplikasi untuk menyediakan input audio sendiri ke SDK siaran, alih-alih terbatas pada mikrofon bawaan perangkat. Sumber audio khusus memungkinkan aplikasi untuk mengalirkan audio yang diproses dengan efek, mencampur beberapa aliran audio, atau mengintegrasikan dengan pustaka pemrosesan audio pihak ketiga.

Saat Anda menggunakan sumber input audio khusus, SDK siaran tidak lagi bertanggung jawab untuk mengelola mikrofon secara langsung. Sebagai gantinya, aplikasi Anda bertanggung jawab untuk menangkap, memproses, dan mengirimkan data audio ke sumber khusus.

`custom-audio-source` Alur kerja mengikuti langkah-langkah ini:

1. Input audio - Buat sumber audio khusus dengan format audio tertentu (laju sampel, saluran, format).
2. Pemrosesan Anda — Tangkap atau hasilkan data audio dari pipeline pemrosesan audio Anda.
3. Sumber audio khusus - Kirim buffer audio ke sumber khusus menggunakan `appendBuffer()`.
4. Panggung — Bungkus `LocalStageStream` dan publikasikan ke panggung melalui `AndaStageStrategy`.
5. Peserta — Peserta panggung menerima audio yang diproses secara real time.

Android

Membuat Sumber Audio Kustom

Setelah Anda membuat `DeviceDiscovery` sesi, buat sumber input audio kustom:

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

// Create custom audio source with specific format
CustomAudioSource customAudioSource = deviceDiscovery.createAudioInputSource(
    2, // Number of channels (1 = mono, 2 = stereo)
    BroadcastConfiguration.AudioSampleRate.RATE_48000, // Sample rate
```

```
AudioDevice.Format.INT16 // Audio format (16-bit PCM)
);
```

Metode ini mengembalikan `CustomAudioSource`, yang menerima data audio PCM mentah. Sumber audio kustom harus dikonfigurasi dengan format audio yang sama dengan yang dihasilkan pipeline pemrosesan audio Anda.

Format Audio yang Didukung

Parameter	Opsi	Deskripsi
Saluran	1 (mono), 2 (stereo)	Jumlah saluran audio.
Tingkat sampel	RATE_16000, RATE_44100, RATE_48000	Kecepatan sampel audio dalam Hz. 48kHz direkomendasikan untuk kualitas tinggi.
Format	INT16, FLOAT32	Format sampel audio. INT16 adalah PCM titik tetap 16-bit, adalah PCM floating-point FLOAT32 32-bit. Format interleaved dan planar tersedia.

Mengirimkan Data Audio

Untuk mengirimkan data audio ke sumber kustom, gunakan `appendBuffer()` metode ini:

```
// Prepare audio data in a ByteBuffer
ByteBuffer audioBuffer = ByteBuffer.allocateDirect(bufferSize);
audioBuffer.put(pcmAudioData); // Your processed audio data

// Calculate the number of bytes
long byteCount = pcmAudioData.length;

// Submit audio to the custom source
// presentationTimeUs should be generated by and come from your audio source
int samplesProcessed = customAudioSource.appendBuffer(
    audioBuffer,
    byteCount,
    presentationTimeUs
);
```

```

if (samplesProcessed > 0) {
    Log.d(TAG, "Successfully submitted " + samplesProcessed + " samples");
} else {
    Log.w(TAG, "Failed to submit audio samples");
}

// Clear buffer for reuse
audioBuffer.clear();

```

Pertimbangan penting:

- Data audio harus dalam format yang ditentukan saat membuat sumber khusus.
- Stempel waktu harus ditingkatkan secara monoton dan disediakan oleh sumber audio Anda untuk pemutaran audio yang lancar.
- Kirim audio secara teratur untuk menghindari celah dalam aliran.
- Metode mengembalikan jumlah sampel yang diproses (0 menunjukkan kegagalan).

Penerbitan ke Panggung

Bungkus CustomAudioSource dalam AudioLocalStageStream dan kembalikan dariStageStrategy:

```

// Create the audio stream from custom source
AudioLocalStageStream audioStream = new AudioLocalStageStream(customAudioSource);

// Define your stage strategy
Strategy stageStrategy = new Strategy() {
    @NonNull
    @Override
    public List<LocalStageStream> stageStreamsToPublishForParticipant(
        @NonNull Stage stage,
        @NonNull ParticipantInfo participantInfo) {
        List<LocalStageStream> streams = new ArrayList<>();
        streams.add(audioStream); // Publish custom audio
        return streams;
    }

    @Override
    public boolean shouldPublishFromParticipant(
        @NonNull Stage stage,
        @NonNull ParticipantInfo participantInfo) {

```

```
        return true; // Control when to publish
    }

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

// Create and join the stage
Stage stage = new Stage(context, stageToken, stageStrategy);
```

Contoh Lengkap: Integrasi Pemrosesan Audio

Berikut adalah contoh lengkap yang menunjukkan integrasi dengan SDK pemrosesan audio:

```
public class AudioStreamingActivity extends AppCompatActivity {
    private DeviceDiscovery deviceDiscovery;
    private CustomAudioSource customAudioSource;
    private AudioLocalStageStream audioStream;
    private Stage stage;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Configure audio manager
        StageAudioManager.getInstance(this)
            .setPreset(StageAudioManager.UseCasePreset.VIDEO_CHAT);

        // Initialize IVS components
        initializeIVSStage();

        // Initialize your audio processing SDK
        initializeAudioProcessing();
    }

    private void initializeIVSStage() {
        deviceDiscovery = new DeviceDiscovery(this);

        // Create custom audio source (48kHz stereo, 16-bit)
```

```
customAudioSource = deviceDiscovery.createAudioInputSource(
    2, // Stereo
    BroadcastConfiguration.AudioSampleRate.RATE_48000,
    AudioDevice.Format.INT16
);

// Create audio stream
audioStream = new AudioLocalStageStream(customAudioSource);

// Create stage with strategy
Strategy strategy = new Strategy() {
    @NonNull
    @Override
    public List<LocalStageStream> stageStreamsToPublishForParticipant(
        @NonNull Stage stage,
        @NonNull ParticipantInfo participantInfo) {
        return Collections.singletonList(audioStream);
    }

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

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

stage = new Stage(this, getStageToken(), strategy);
}

private void initializeAudioProcessing() {
    // Initialize your audio processing SDK
    // Set up callback to receive processed audio
    yourAudioSDK.setAudioCallback(new AudioCallback() {
        @Override
        public void onProcessedAudio(byte[] audioData, int sampleRate,
            int channels, long timestamp) {
```

```
        // Submit processed audio to IVS Stage
        submitAudioToStage(audioData, timestamp);
    }
});
}

// The timestamp is required to come from your audio source and you
// should not be generating one on your own, unless your audio source
// does not provide one. If that is the case, create your own epoch
// timestamp and manually calculate the duration between each sample
// using the number of frames and frame size.

private void submitAudioToStage(byte[] audioData, long timestamp) {
    try {
        // Allocate direct buffer
        ByteBuffer buffer = ByteBuffer.allocateDirect(audioData.length);
        buffer.put(audioData);

        // Submit to custom audio source
        int samplesProcessed = customAudioSource.appendBuffer(
            buffer,
            audioData.length,
            timestamp > 0 ? timestamp : System.nanoTime() / 1000
        );

        if (samplesProcessed <= 0) {
            Log.w(TAG, "Failed to submit audio samples");
        }

        buffer.clear();
    } catch (Exception e) {
        Log.e(TAG, "Error submitting audio: " + e.getMessage(), e);
    }
}

@Override
protected void onDestroy() {
    super.onDestroy();
    if (stage != null) {
        stage.release();
    }
}
}
```

Praktik Terbaik

Konsistensi Format Audio

Pastikan format audio yang Anda kirimkan cocok dengan format yang ditentukan saat membuat sumber kustom:

```
// If you create with 48kHz stereo INT16
customAudioSource = deviceDiscovery.createAudioInputSource(
    2, RATE_48000, INT16
);

// Your audio data must be:
// - 2 channels (stereo)
// - 48000 Hz sample rate
// - 16-bit interleaved PCM format
```

Manajemen Buffer

Gunakan langsung `ByteBuffer`s dan gunakan kembali untuk meminimalkan pengumpulan sampah:

```
// Allocate once
private ByteBuffer audioBuffer = ByteBuffer.allocateDirect(BUFFER_SIZE);

// Reuse in callback
public void onAudioData(byte[] data) {
    audioBuffer.clear();
    audioBuffer.put(data);
    customAudioSource.appendBuffer(audioBuffer, data.length, getTimestamp());
    audioBuffer.clear();
}
```

Pengaturan Waktu dan Sinkronisasi

Anda harus menggunakan stempel waktu yang disediakan oleh sumber audio Anda untuk pemutaran audio yang lancar. Jika sumber audio Anda tidak menyediakan stempel waktu sendiri, buat stempel waktu epoch Anda sendiri dan hitung durasi antara setiap sampel secara manual menggunakan jumlah bingkai dan ukuran bingkai.

```
// "audioFrameTimestamp" should be generated by your audio source
// Consult your audio source's documentation for information on how to get this
```

```
long timestamp = audioFrameTimestamp;
```

Penanganan Kesalahan

Selalu periksa nilai pengembalian `appendBuffer()`:

```
int samplesProcessed = customAudioSource.appendBuffer(buffer, count, timestamp);

if (samplesProcessed <= 0) {
    Log.w(TAG, "Audio submission failed - buffer may be full or format mismatch");
    // Handle error: check format, reduce submission rate, etc.
}
```

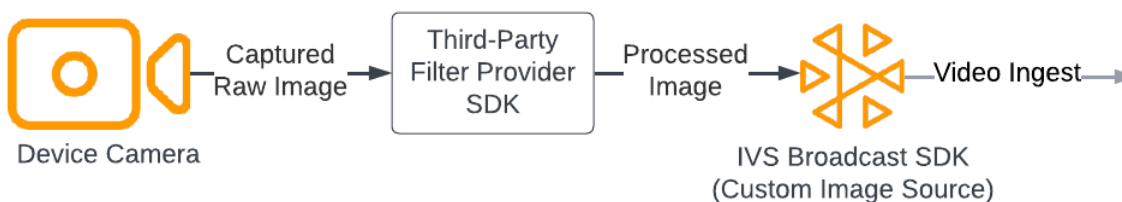
SDK Siaran IVS: Filter Kamera Pihak Ketiga | Streaming Waktu Nyata

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 filter kamera pihak ketiga SDKs 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 a [LocalStageStream](#) dan dipublikasikan ke Stage.
- 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.

Menggunakan BytePlus dengan IVS Broadcast SDK

Dokumen ini menjelaskan cara menggunakan BytePlus Effects SDK dengan IVS broadcast SDK.

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 [GLSurfaceView.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);
```

Menggunakan DeepAR dengan IVS Broadcast SDK

Dokumen ini menjelaskan cara menggunakan DeepAR SDK dengan IVS broadcast SDK.

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.

Menggunakan Snap dengan IVS Broadcast SDK

Dokumen ini menjelaskan cara menggunakan SDK Kit Kamera Snap dengan SDK siaran IVS.

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. Buat `index.css`.
5. Tampilkan dan atur peserta.
6. Tampilkan kamera dan mikrofon yang terhubung.
7. Buat sesi Kit Kamera.
8. Ambil lensa dan isi pemilih lensa.
9. Render output dari sesi Kit Kamera ke kanvas.
10. Buat fungsi untuk mengisi dropdown Lensa.

11.Menyediakan Kit Kamera dengan sumber media untuk rendering dan mempublikasikan file.

LocalStageStream

12.Buatpackage.json.

13.Buat file konfigurasi Webpack.

14.Siapkan server HTTPS dan uji.

Masing-masing langkah ini dijelaskan di bawah ini.

Instal SDK Kit Kamera dan Webpack

Dalam contoh ini kami menggunakan Webpack sebagai bundler kami; Namun, Anda dapat menggunakan bundler apa pun.

```
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.

HTML

```
<!--  
/!* 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/LowLatencyUserGuide/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 -->

  <!-- Display Local Participants -->

  <!-- Lens Selector -->

  <!-- Display Remote Participants -->

  <!-- Load All Desired Scripts -->

```

Tambahkan Elemen Pengaturan

Buat HTML untuk memilih kamera, mikrofon, dan lensa dan tentukan token peserta:

HTML

```

<!-- 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 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:

HTML

```

<!-- 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.) Tambahkan tag penutup untuk HTML elemen body dan untuk menyelesaikan pembuatan `index.html`.

HTML

```
<!-- 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>
</body>
</html>
```

Buat index.css

Buat file sumber CSS untuk menata halaman. Kami tidak akan membahas kode ini untuk fokus pada logika untuk mengelola Stage dan mengintegrasikan dengan Snap Camera Kit SDK.

CSS

```
/*! Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved. SPDX-License-
Identifier: Apache-2.0 */

html,
body {
  margin: 2rem;
  box-sizing: border-box;
  height: 100vh;
  max-height: 100vh;
  display: flex;
  flex-direction: column;
}
```

```
hr {
  margin: 1rem 0;
}

table {
  display: table;
}

canvas {
  margin-bottom: 1rem;
  background: green;
}

video {
  margin-bottom: 1rem;
  background: black;
  max-width: 100%;
  max-height: 150px;
}

.log {
  flex: none;
  height: 300px;
}

.content {
  flex: 1 0 auto;
}

.button {
  display: block;
  margin: 0 auto;
}

.local-container {
  position: relative;
}

.static-controls {
  position: absolute;
  margin-left: auto;
  margin-right: auto;
  left: 0;
```

```
    right: 0;
    bottom: -4rem;
    text-align: center;
}

.static-controls button {
  display: inline-block;
}

.hidden {
  display: none;
}

.participant-container {
  display: flex;
  align-items: center;
  justify-content: center;
  flex-direction: column;
  margin: 1rem;
}

video {
  border: 0.5rem solid #555;
  border-radius: 0.5rem;
}

.placeholder {
  background-color: #333333;
  display: flex;
  text-align: center;
  margin-bottom: 1rem;
}

.placeholder span {
  margin: auto;
  color: white;
}

#local-media {
  display: inline-block;
  width: 100vw;
}

#local-media video {
  max-height: 300px;
}
```

```
#remote-media {
  display: flex;
  justify-content: center;
  align-items: center;
  flex-direction: row;
  width: 100%;
}

#lens-selector {
  width: 100%;
  margin-bottom: 1rem;
}
```

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. Kami merekomendasikan untuk menyalin dan menempelkan blok kode berikut ke dalam `stages.js` Anda kemudian dapat meninjau kode sepotong demi sepotong untuk memahami apa yang terjadi di bagian berikut.

JavaScript

```
/*! Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved. SPDX-License-
Identifier: Apache-2.0 */

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');

let lensSelector = document.getElementById('lens-selector');
let session;
let availableLenses = [];

// 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_YOUR_API_TOKEN_HERE',
  });

  session = await cameraKit.createSession({ liveRenderTarget });
  const { lenses } = await cameraKit.lensRepository.loadLensGroups([
    'INSERT_YOUR_LENS_GROUP_ID_HERE',
  ]);

  availableLenses = lenses;
  populateLensSelector(lenses);

  const snapStream = liveRenderTarget.captureStream();

  lensSelector.addEventListener('change', handleLensChange);
  lensSelector.disabled = true;
  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();
  });
};

async function setCameraKitSource(session, mediaStream) {
  const source = createMediaStreamSource(mediaStream);
  await session.setSource(source);
  source.setTransform(Transform2D.MirrorX);
  session.play();
}

const populateLensSelector = (lenses) => {
  lensSelector.innerHTML = '<option selected disabled>Choose Lens</option>';

  lenses.forEach((lens, index) => {
    const option = document.createElement('option');
    option.value = index;
    option.text = lens.name || `Lens ${index + 1}`;
    lensSelector.appendChild(option);
  });
};

const handleLensChange = (event) => {
  const selectedIndex = parseInt(event.target.value);
  if (session && availableLenses[selectedIndex]) {
    session.applyLens(availableLenses[selectedIndex]);
  }
};

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(snapStream.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');
    lensSelector.disabled = false;
  } else {
    controls.classList.add('hidden');
    lensSelector.disabled = true;
  }
});
```

```
    }
  });

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();
```

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.

JavaScript

```
/*! Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved. SPDX-License-
Identifier: Apache-2.0 */

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');

let lensSelector = document.getElementById('lens-selector');
let session;
let availableLenses = [];

// 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_YOUR_API_TOKEN_HERE',
  });

  session = await cameraKit.createSession({ liveRenderTarget });
```

Ambil Lensa dan Isi Pemilih Lensa

Untuk mengambil Lensa Anda, ganti placeholder untuk ID Grup Lensa dengan milik Anda sendiri, yang dapat ditemukan di Portal Pengembang [Kit Kamera](#). Isi dropdown pemilihan Lensa menggunakan `populateLensSelector()` fungsi yang akan kita buat nanti.

JavaScript

```
session = await cameraKit.createSession({ liveRenderTarget });
const { lenses } = await cameraKit.lensRepository.loadLensGroups([
  'INSERT_YOUR_LENS_GROUP_ID_HERE',
]);

availableLenses = lenses;
populateLensSelector(lenses);
```

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();

lensSelector.addEventListener('change', handleLensChange);
lensSelector.disabled = true;
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();
});
};
```

Buat Fungsi untuk Mengisi Dropdown Lensa

Buat fungsi berikut untuk mengisi pemilih Lensa dengan lensa yang diambil sebelumnya. Pemilih Lensa adalah elemen UI pada halaman yang memungkinkan Anda memilih dari daftar lensa untuk diterapkan ke umpan kamera. Juga, buat fungsi `handleLensChange` callback untuk menerapkan lensa yang ditentukan saat dipilih dari dropdown Lens.

JavaScript

```
const populateLensSelector = (lenses) => {
  lensSelector.innerHTML = '<option selected disabled>Choose Lens</option>';

  lenses.forEach((lens, index) => {
    const option = document.createElement('option');
    option.value = index;
    option.text = lens.name || `Lens ${index + 1}`;
    lensSelector.appendChild(option);
  });
};

const handleLensChange = (event) => {
  const selectedIndex = parseInt(event.target.value);
  if (session && availableLenses[selectedIndex]) {
    session.applyLens(availableLenses[selectedIndex]);
  }
};
```

Menyediakan Kit Kamera dengan Sumber Media untuk Rendering dan Publikasikan LocalStageStream

Untuk mempublikasikan aliran video dengan Lens 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 (via `LocalCamera`) dan objek sesi ke `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();
```

Buat package.json

Buat package.json dan tambahkan konfigurasi JSON berikut. File ini mendefinisikan dependensi kita dan menyertakan perintah script untuk bundling kode kita.

Konfigurasi JSON

```
{
  "dependencies": {
    "@snap/camera-kit": "^0.10.0"
  },
  "name": "ivs-stages-with-snap-camerakit",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
```

```
    "build": "webpack"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": "",
  "devDependencies": {
    "webpack": "^5.95.0",
    "webpack-cli": "^5.1.4"
  }
}
```

Membuat File Konfigurasi Webpack

Buat `webpack.config.js` dan tambahkan kode berikut. Ini menggabungkan kode yang kami buat sejauh ini sehingga kami 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. Untuk tujuan pengujian, Anda kemudian dapat melayani HTML dan JavaScript dari komputer lokal Anda. Dalam contoh ini, kita menggunakan `http.server` modul Python.

Siapkan Server HTTPS dan Uji

Untuk menguji kode kita, kita perlu menyiapkan server HTTPS. Menggunakan server HTTPS untuk pengembangan lokal dan pengujian integrasi aplikasi web Anda dengan Snap Camera Kit SDK akan membantu menghindari masalah CORS (Cross-Origin Resource Sharing).

Buka terminal dan arahkan ke direktori tempat Anda membuat semua kode hingga saat ini. Jalankan perintah berikut untuk menghasilkan sertifikat SSL/TLS yang ditandatangani sendiri dan kunci pribadi:

```
openssl req -x509 -newkey rsa:4096 -keyout key.pem -out cert.pem -days 365 -nodes
```

Ini menciptakan dua file: `key.pem` (kunci pribadi) dan `cert.pem` (sertifikat yang ditandatangani sendiri). Buat file Python baru bernama `https_server.py` dan tambahkan kode berikut:

Python

```
import http.server
import ssl

# Set the directory to serve files from
DIRECTORY = '.'

# Create the HTTPS server
server_address = ('', 4443)
httpd = http.server.HTTPServer(
    server_address, http.server.SimpleHTTPRequestHandler)

# Wrap the socket with SSL/TLS
context = ssl.SSLContext(ssl.PROTOCOL_TLS_SERVER)
context.load_cert_chain('cert.pem', 'key.pem')
httpd.socket = context.wrap_socket(httpd.socket, server_side=True)

print(f'Starting HTTPS server on https://localhost:4443, serving {DIRECTORY}')
httpd.serve_forever()
```

Buka terminal, arahkan ke direktori tempat Anda membuat `https_server.py` file, dan jalankan perintah berikut:

```
python3 https_server.py
```

Ini memulai server HTTPS di `https://localhost:4443`, menyajikan file dari direktori saat ini. Pastikan `key.pem` file `cert.pem` dan file berada di direktori yang sama dengan `https_server.py` file.

Buka browser Anda dan arahkan ke `https://localhost:4443`. Karena ini adalah SSL/TLS sertifikat yang ditandatangani sendiri, itu tidak akan dipercaya oleh browser web Anda, jadi Anda akan mendapatkan peringatan. Karena ini hanya untuk tujuan pengujian, Anda dapat melewati peringatan. Anda kemudian akan melihat efek AR untuk Lensa Snap yang Anda tentukan sebelumnya diterapkan ke umpan kamera Anda di layar.

Perhatikan bahwa pengaturan ini menggunakan built-in `http.server` dan `ssl` modul Python cocok untuk tujuan pengembangan dan pengujian lokal, tetapi tidak disarankan untuk lingkungan produksi. SSL/TLS Sertifikat yang ditandatangani sendiri yang digunakan dalam pengaturan ini tidak dipercaya oleh browser web dan klien lain, yang berarti pengguna akan menghadapi peringatan keamanan saat mengakses server. Selain itu, meskipun kami menggunakan modul `http.server` dan `ssl` bawaan Python dalam contoh ini, Anda dapat memilih untuk menggunakan solusi server HTTPS lain.

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 `cameraKitSession` Camera Kit juga menyediakan pembungkus yang nyaman untuk [APIsCameraX](#) 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 olehSurfaceSource.

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)

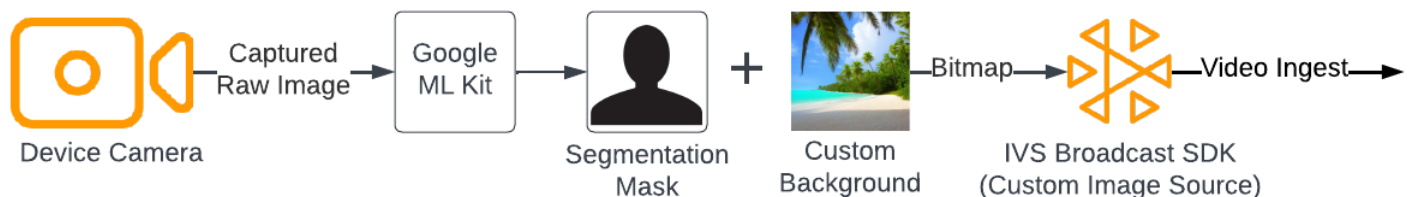
@Override
fun stageStreamsToPublishForParticipant(stage: Stage, participantInfo:
ParticipantInfo): List<LocalStageStream> = publishStreams

```

Menggunakan Penggantian Latar Belakang dengan SDK Siaran IVS

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 streaming langsung dengan gambar khusus, 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 `bernamasegmentationStream`. 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 `aLocalStageStream`, 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 bernama `initBackgroundCanvas`, yang memuat gambar kustom dari file lokal dan merendernya ke kanvas. Kami akan mendefinisikan fungsi ini pada langkah berikutnya. Tetapkan `MediaStream` diambil dari kamera pengguna ke objek video. Nantinya, objek video ini akan diteruskan ke `ImageSegmenter`. 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 `instanceImageSegmenter`, yang akan mengelompokkan gambar dan mengembalikan hasilnya sebagai topeng. Saat membuat `instanceImageSegmenter`, 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_segmenter/selfie_segmenter/float16/latest/selfie_segmenter.tflite",
      delegate: "GPU",
    },
    runningMode: "VIDEO",
    outputCategoryMask: true,
  });
};
```

```
};
```

Render Umpan Video ke Kanvas

Selanjutnya, buat fungsi yang membuat 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 `kamiImageSegmenter`, 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_segmenter/selfie_segmenter/float16/latest/selfie_segmenter.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 impor 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 khusus 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 diri, 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 ada 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, ponsel kami SDKs menyediakan beberapa konfigurasi audio preset, serta penyesuaian yang lebih kuat sesuai kebutuhan.

Pengantar

Siaran seluler IVS SDKs menyediakan `StageAudioManager` kelas. 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 [AVAudioSesi](#) aplikasi, 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.

Kategori Volume Rocker mengacu pada jenis volume (volume media atau volume panggilan) yang digunakan atau diubah melalui rocker volume fisik pada perangkat. Perhatikan bahwa ini memengaruhi volume saat beralih mode audio. Misalnya, volume perangkat diatur ke nilai maksimum saat menggunakan preset Obrolan Video. Beralih ke preset Subscribe Only menyebabkan tingkat volume yang berbeda dari sistem operasi, yang dapat menyebabkan perubahan volume yang signifikan pada perangkat.

Obrolan Video

Ini adalah preset default, yang dirancang untuk saat perangkat lokal akan melakukan percakapan real-time dengan peserta lain.

Masalah yang diketahui di iOS: Menggunakan preset ini dan tidak memasang mikrofon menyebabkan audio diputar melalui lubang suara alih-alih speaker perangkat. Gunakan preset ini hanya dalam kombinasi dengan mikrofon.

Kategori	Android	iOS
Pembatalan Echo	Diaktifkan	Diaktifkan
Volume Rocker	Volume Panggilan	Volume Panggilan
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.

Kategori	Android	iOS
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 ketika 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.
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	Pembatalan gema platform dinonaktifkan, tetapi pembatalan gema perangkat lunak mungkin masih terjadi jika benar. <code>StageAudioConfiguration.enableEchoCancellation</code>	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	Tinggi. Kedua belah pihak harus dapat mengirim musik dan mendengarnya dengan jelas di sisi lain. Saat headset Bluetooth terhubung, kualitas audio akan turun karena mode Bluetooth SCO diaktifkan.	Tinggi. Kedua belah pihak harus dapat mengirim musik dan mendengarnya dengan jelas di sisi lain. Saat headset Bluetooth terhubung, kualitas audio mungkin turun karena mode Bluetooth SCO diaktifkan, tergantung pada headset.

Kasus Penggunaan Tingkat Lanjut

Di luar preset, baik siaran streaming real-time iOS dan Android SDKs memungkinkan konfigurasi mode audio platform yang mendasarinya:

- Di Android, atur [AudioSource](#), [Penggunaan](#), dan [ContentType](#).
- Di iOS, gunakan [AVAudioSession.Category](#), [Session. AVAudio CategoryOptions](#), [AVAudioSession.Mode](#), dan kemampuan untuk beralih jika [pemrosesan suara](#) diaktifkan atau tidak saat penerbitan.

Catatan: Saat menggunakan metode SDK audio ini, dimungkinkan untuk salah mengonfigurasi sesi audio yang mendasarinya. Misalnya, menggunakan `.allowBluetooth` opsi di iOS dalam kombinasi dengan `.playback` kategori membuat konfigurasi audio yang tidak valid dan SDK tidak dapat merekam atau memutar audio. Metode ini dirancang untuk digunakan hanya ketika aplikasi memiliki persyaratan sesi audio tertentu yang telah divalidasi.

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
```

Pembatalan Echo iOS

Pembatalan gema di iOS dapat dikontrol secara independen melalui `IVSStageAudioManager` juga menggunakan `metodenyaechoCancellationEnabled`. Metode ini mengontrol apakah [pemrosesan suara](#) diaktifkan pada node input dan output yang mendasari yang `AVAudioEngine` digunakan oleh SDK. Penting untuk memahami efek mengubah properti ini secara manual:

- `AVAudioEngine` properti hanya dihormati jika mikrofon SDK aktif; ini diperlukan karena persyaratan iOS bahwa pemrosesan suara diaktifkan pada node input dan output secara bersamaan. Biasanya ini dilakukan dengan menggunakan mikrofon yang dikembalikan `IVSDeviceDiscovery` untuk membuat `IVSLocalStageStream` untuk menerbitkan. Sebagai alternatif, mikrofon dapat diaktifkan, tanpa digunakan untuk mempublikasikan, dengan melampirkan `IVSAudioDeviceStatsCallback` ke mikrofon itu sendiri. Pendekatan alternatif ini berguna jika pembatalan gema diperlukan saat menggunakan mikrofon khusus alih-alih `audio-source-based` mikrofon IVS SDK.
- Mengaktifkan `AVAudioEngine` properti membutuhkan mode `.videoChat` atau `.voiceChat`. Meminta mode yang berbeda menyebabkan kerangka audio dasar iOS melawan SDK, menyebabkan kehilangan audio.
- Mengaktifkan `AVAudioEngine` secara otomatis mengaktifkan `.allowBluetooth` opsi.

Perilaku dapat berbeda tergantung pada perangkat dan versi iOS.

Sumber Audio Kustom iOS

Sumber audio khusus dapat digunakan dengan SDK dengan menggunakan `IVSDeviceDiscovery.createAudioSource`. Saat menghubungkan ke Stage, SDK siaran streaming real-time IVS masih mengelola `AVAudioEngine` instans internal untuk pemutaran audio, bahkan jika mikrofon SDK tidak digunakan. Akibatnya, nilai yang diberikan `IVSStageAudioManager` harus kompatibel dengan audio yang disediakan oleh sumber audio khusus.

Jika sumber audio kustom yang digunakan untuk mempublikasikan merekam dari mikrofon tetapi dikelola oleh aplikasi host, SDK pembatalan gema di atas tidak akan berfungsi kecuali mikrofon yang dikelola SDK diaktifkan. Untuk mengatasi persyaratan tersebut, lihat [Pembatalan Gema iOS](#).

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 Lainnya SDKs

Karena iOS dan Android hanya mendukung satu mode audio aktif per aplikasi, biasanya terjadi konflik jika aplikasi Anda menggunakan beberapa SDKs 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, keduanya SDKs 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 peristiwa 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 menampilkan ke Destination gagal (misalnya, bucket S3 tidak ditemukan, akses ditolak ke bucket S3, atau aliran sudah ada untuk tujuan RTMP).
Perubahan Status Komposisi IVS	Tujuan Mulai	Output ke Destination berhasil dimulai.

Jenis Acara	Peristiwa	Dikirim Ketika...
Perubahan Status Komposisi IVS	Tujuan Akhir	Output ke Tujuan selesai.
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 karena tahap dihapus, satu atau lebih output gagal, atau kesalahan internal lainnya.
Perubahan Status Rekaman Peserta IVS	Merekam Mulai	Penerbit telah terhubung ke panggung dan sedang direkam ke S3.
Perubahan Status Rekaman Peserta IVS	Rekaman Akhir	Penerbit telah terputus dari panggung dan semua file yang tersisa telah ditulis ke S3.
Perubahan Status Rekaman Peserta IVS	Merekam Kegagalan Mulai	Penerbit terhubung ke panggung, tetapi perekaman gagal dimulai karena kesalahan (misalnya, jika bucket S3 tidak ditemukan atau tidak dapat diakses). Streaming langsung penerbit ini tidak direkam.

Jenis Acara	Peristiwa	Dikirim Ketika...
Perubahan Status Rekaman Peserta IVS	Merekam Kegagalan Akhir	Perekaman berakhir dengan kegagalan, karena kesalahan yang ditemui selama perekaman (misalnya , jika bucket S3 tidak ditemukan atau tidak dapat diakses). Beberapa objek mungkin masih ditulis ke lokasi penyimpanan yang dikonfigurasi.
Pembaruan Tahap IVS	Peserta Diterbitkan	Seorang peserta mulai menerbitkan ke panggung.
Pembaruan Tahap IVS	Peserta Tidak Dipublikasikan	Seorang peserta telah berhenti menerbitkan ke panggung.
Pembaruan Tahap IVS	Kesalahan Publikasikan Peserta	Upaya peserta untuk mempublikasikan ke tahap gagal.
Pembaruan Tahap IVS	Replikasi Peserta Mulai	Replikasi peserta dimulai.
Pembaruan Tahap IVS	Akhir Replikasi Peserta	Replikasi peserta berakhir. Replikasi dapat berakhir karena operasi StopParticipantReplication API, jika penerbit telah berhenti menerbitkan, atau jika penerbit telah berhenti menerbitkan dan jendela penyambungan kembali telah kedaluwarsa.
Pembaruan Tahap IVS	Token dipertukarkan	Token peserta yang ada ditukar dengan yang baru. Pertukaran ini menghasilkan peningkatan atau penurunan kemampuan token atribut token yang and/or diperbarui.

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 menampilkan ke Destination gagal (misalnya, bucket S3 tidak ditemukan, akses ditolak ke bucket S3, atau aliran sudah ada untuk tujuan RTMP).

```
{
  "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>",
    "error_code": "e.g., AccessDeniedException",
    "reason": "e.g., Access denied to S3 bucket. Please verify your bucket policy"
  }
}
```

Daftar tabel berikut `error_code` dan `reason` nilai untuk peristiwa Kegagalan Tujuan, bersama dengan panduan pemecahan masalah:

error_code	akal budi	Panduan Pemecahan Masalah
ResourceNotFoundException	Ember S3 tidak ditemukan.	Pastikan bucket S3 Anda ada dan berada di wilayah yang benar.

error_code	akal budi	Panduan Pemecahan Masalah
	Harap verifikasi ember Anda ada.	
AccessDeniedException	Akses ditolak ke bucket S3. Harap verifikasi kebijakan bucket Anda.	Verifikasi kebijakan bucket S3 Anda memberi layanan IVS izin yang diperlukan.
ConflictException	Stream sudah ada	Pastikan tidak ada siaran lain yang aktif di saluran tujuan RTMP yang sama.
InternalServerErrorException	Kesalahan internal layanan	Coba lagi operasi. Jika masalah berlanjut, hubungi AWS Support.

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: Peristiwa ini dikirim ketika pipeline Komposisi gagal karena tahap dihapus, satu atau beberapa output gagal, 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>",
  "error_code": "e.g., DestinationFailure",
  "reason": "e.g. One or more outputs failed"
}
}

```

Daftar tabel berikut `error_code` dan `reason` nilai untuk peristiwa Kegagalan Sesi, bersama dengan panduan pemecahan masalah:

error_code	akal budi	Panduan Pemecahan Masalah
StageDeleted	Tahap telah dihapus	Verifikasi tahapan ada sebelum memulai komposisi.
DestinationFailure	Satu atau lebih output gagal	Periksa kesalahan tujuan individu.
InternalServerErrorException	Kesalahan internal layanan	Coba lagi operasi. Jika masalah berlanjut, hubungi AWS Support.

Contoh: Perubahan Status Pencatatan Peserta Individu

Merekam Mulai: Acara ini dikirim ketika penerbit telah terhubung ke panggung dan sedang direkam ke S3.

```

{
  "version": "0",
  "id": "12345678-1a23-4567-a1bc-1a2b34567890",
  "detail-type": "IVS Participant Recording State Change",
  "source": "aws.ivs",

```

```

"account": "123456789012",
"time": "2024-03-13T22:09:58Z",
"region": "us-east-1",
"resources": ["arn:aws:ivs:us-west-2:aws_account_id:stage/AbCdef1G2hij"],
"detail": {
  "session_id": "st-ZyXwvu1T2s",
  "event_name": "Recording Start",
  "participant_id": "xYz1c2d3e4f",
  "recording_s3_bucket_name": "bucket-name",
  "recording_s3_key_prefix": "<stage_id>/<session_id>/
<participant_id>/2024-01-01T12-00-55Z"
}
}

```

Recording End: Acara ini dikirim ketika penerbit telah terputus dari panggung dan semua file yang tersisa telah ditulis ke S3.

```

{
  "version": "0",
  "id": "12345678-1a23-4567-a1bc-1a2b34567890",
  "detail-type": "IVS Participant Recording State Change",
  "source": "aws.ivs",
  "account": "123456789012",
  "time": "2024-03-13T22:19:04Z",
  "region": "us-east-1",
  "resources": ["arn:aws:ivs:us-west-2:aws_account_id:stage/AbCdef1G2hij"],
  "detail": {
    "session_id": "st-ZyXwvu1T2s",
    "event_name": "Recording End",
    "participant_id": "xYz1c2d3e4f",
    "recording_s3_bucket_name": "bucket-name",
    "recording_s3_key_prefix": "<stage_id>/<session_id>/
<participant_id>/2024-01-01T12-00-55Z",
    "recording_duration_ms": 547327
  }
}

```

Kegagalan Mulai Perekaman: Peristiwa ini dikirim ketika penerbit terhubung ke panggung, tetapi perekaman gagal dimulai karena kesalahan (misalnya, jika bucket S3 tidak ditemukan atau tidak dapat diakses). Streaming langsung penerbit tidak direkam.

```

{

```

```

"version": "0",
"id": "12345678-1a23-4567-a1bc-1a2b34567890",
"detail-type": "IVS Participant Recording State Change",
"source": "aws.ivs",
"account": "123456789012",
"time": "2024-03-13T22:09:58Z",
"region": "us-east-1",
"resources": ["arn:aws:ivs:us-west-2:aws_account_id:stage/AbCdef1G2hij"],
"detail": {
  "session_id": "st-ZyXwvu1T2s",
  "event_name": "Recording Start Failure",
  "participant_id": "xYz1c2d3e4f",
  "recording_s3_bucket_name": "bucket-name",
  "recording_s3_key_prefix": "<stage_id>/<session_id>/
<participant_id>/2024-01-01T12-00-55Z",
  "error_code": "e.g., AccessDeniedException",
  "reason": "e.g., Access denied to S3 bucket. Please verify your bucket policy"
}
}

```

Daftar tabel berikut `error_code` dan `reason` nilai untuk Merekam peristiwa Kegagalan Mulai, bersama dengan panduan pemecahan masalah:

error_code	akal budi	Panduan Pemecahan Masalah
ResourceNotFoundException	Ember S3 tidak ditemukan. Harap verifikasi ember Anda ada.	Pastikan bucket S3 Anda ada dan berada di wilayah yang benar.
AccessDeniedException	Akses ditolak ke bucket S3. Harap verifikasi kebijakan bucket Anda.	Verifikasi kebijakan bucket S3 Anda memberi layanan IVS izin yang diperlukan.
ValidationException	Codec video tidak didukung untuk perekaman	Verifikasi bahwa penerbit menggunakan codec video yang didukung.

error_code	akal budi	Panduan Pemecahan Masalah
InternalServerErrorException	Kesalahan internal layanan	Coba lagi operasi. Jika masalah berlanjut, hubungi AWS Support.

Kegagalan Akhir Perekaman: Peristiwa ini dikirim saat perekaman berakhir dengan kegagalan, karena kesalahan yang ditemui selama perekaman (misalnya, jika bucket S3 tidak ditemukan atau tidak dapat diakses). Beberapa objek mungkin masih ditulis ke lokasi penyimpanan yang dikonfigurasi.

```
{
  "version": "0",
  "id": "12345678-1a23-4567-a1bc-1a2b34567890",
  "detail-type": "IVS Participant Recording State Change",
  "source": "aws.ivs",
  "account": "123456789012",
  "time": "2024-03-13T22:19:04Z",
  "region": "us-east-1",
  "resources": ["arn:aws:ivs:us-west-2:aws_account_id:stage/AbCdef1G2hij"],
  "detail": {
    "session_id": "st-ZyXwvu1T2s",
    "event_name": "Recording End Failure",
    "participant_id": "xYz1c2d3e4f",
    "recording_s3_bucket_name": "bucket-name",
    "recording_s3_key_prefix": "<stage_id>/<session_id>/<participant_id>/2024-01-01T12-00-55Z",
    "recording_duration_ms": 547327,
    "error_code": "e.g., AccessDeniedException",
    "reason": "e.g., Access denied to S3 bucket. Please verify your bucket policy"
  }
}
```

Daftar tabel berikut `error_code` dan `reason` nilai untuk peristiwa Kegagalan Akhir Perekaman, bersama dengan panduan pemecahan masalah:

error_code	akal budi	Panduan Pemecahan Masalah
ResourceNotFoundException	Ember S3 tidak ditemukan.	Pastikan bucket S3 Anda ada dan berada di wilayah yang benar.

error_code	akal budi	Panduan Pemecahan Masalah
	Harap verifikasi ember Anda ada.	
AccessDeniedException	Akses ditolak ke bucket S3. Harap verifikasi kebijakan bucket Anda.	Verifikasi kebijakan bucket S3 Anda memberi layanan IVS izin yang diperlukan.
InternalServerErrorException	Kesalahan internal layanan	Coba lagi operasi. Jika masalah berlanjut, hubungi AWS Support.

Perhatikan bahwa, jika gabungan rekaman peserta individu diaktifkan, dan jika penerbit panggung terputus dari panggung dan kemudian menyambung kembali, IVS mencoba merekam ke awalan S3 yang sama dengan sesi sebelumnya. Akibatnya, dalam contoh di atas, `session_id` komponen `recording_s3_key_prefix` dapat memiliki nilai yang berbeda dari `session_id` bidang `detail`. Lihat [Menggabungkan Rekaman Peserta Individu yang Terfragmentasi](#).

Contoh: Pembaruan Tahap

Acara pembaruan panggung mencakup nama acara (yang mengklasifikasikan acara) dan metadata tentang acara tersebut. Metadata mencakup ID peserta yang memicu peristiwa, tahap dan sesi terkait IDs, 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-ZyXwvu1T2s",
  "event_name": "Participant Published",
  "event_time": "2025-11-18T16:40:32Z",
  "user_id": "Your User Id",
  "participant_id": "xYz1c2d3e4f",
  "replica": true,
  "source_stage_arn": "arn:aws:ivs:us-west-2:123456789012:stage/AbCdef1G2hij",
  "source_session_id": "st-sdfdfdfgdfgh"
}
}

```

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-ZyXwvu1T2s",
    "event_name": "Participant Unpublished",
    "event_time": "2025-11-18T16:40:32Z",
    "user_id": "Your User Id",
    "participant_id": "xYz1c2d3e4f",
    "replica": true,
    "source_stage_arn": "arn:aws:ivs:us-west-2:123456789012:stage/AbCdef1G2hij",
    "source_session_id": "st-sdfdfdfgdfgh"
  }
}

```

Kesalahan Publikasikan Peserta: Acara ini dikirim ketika upaya peserta untuk mempublikasikan ke tahap gagal.

```

{
  "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-ZyXwvu1T2s",
  "event_name": "Participant Publish Error",
  "event_time": "2024-08-13T14:38:17.089061676Z",
  "user_id": "Your User Id",
  "participant_id": "xYz1c2d3e4f",
  "error_code": "BITRATE_EXCEEDED",
  "replica": true,
  "source_stage_arn": "arn:aws:ivs:us-west-2:123456789012:stage/AbCdef1G2hij",
  "source_session_id": "st-sdfdfdfgdfgh"
}
}

```

Mulai Replikasi Peserta: Acara ini dikirim saat replikasi peserta dimulai.

```

{
  "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-ZyXwvu1T2s",
    "event_name": "Participant Replication Start",
    "event_time": "2025-11-18T16:40:32Z",
    "user_id": "Your User Id",
    "participant_id": "xYz1c2d3e4f",
    "destination_stage_arn": "arn:aws:ivs:us-west-2:123456789012:stage/XYZdef1G2hij",
    "destination_session_id": "aBC1c2d3e4f"
  }
}

```

```
}
}
```

Akhir Replikasi Peserta: Acara ini dikirim ketika replikasi peserta berakhir. Replikasi dapat berakhir karena operasi `StopParticipantReplication` API, jika penerbit telah berhenti menerbitkan, atau jika penerbit telah berhenti menerbitkan dan jendela penyambungan kembali telah kedaluwarsa.

```
{
  "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-ZyXwvu1T2s",
    "event_name": "Participant Replication End",
    "event_time": "2025-11-18T16:40:32Z",
    "user_id": "Your User Id",
    "participant_id": "xYz1c2d3e4f",
    "destination_stage_arn": "arn:aws:ivs:us-west-2:123456789012:stage/XYZdef1G2hij",
    "destination_session_id": "aBC1c2d3e4f"
  }
}
```

Token Exchanged: Acara ini dikirim ketika token peserta yang ada ditukar dengan yang baru, sehingga kemampuan token yang ditingkatkan atau diturunkan perbarui atribut token. and/or

```
{
  "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-ZyXwvu1T2s",
    "event_name": "Token Exchanged",
    "event_time": "2025-11-12T20:54:53Z",
    "user_id": "UpdatedUser",
    "participant_id": "xYz1c2d3e4f",
    "previous_token": {
      "capabilities": ["SUBSCRIBE"],
      "attributes": {
        "role": "viewer"
      },
      "user_id": "InitialUser",
      "expiration_time": "2025-11-12T21:54:52Z"
    },
    "new_token": {
      "capabilities": ["SUBSCRIBE", "PUBLISH"],
      "attributes": {
        "role": "moderator"
      },
      "user_id": "UpdatedUser",
      "expiration_time": "2025-11-12T22:54:52Z"
    }
  }
}
```

Komposisi Sisi Server IVS | 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 operasi pesawat kontrol IVS di wilayah asal panggung.

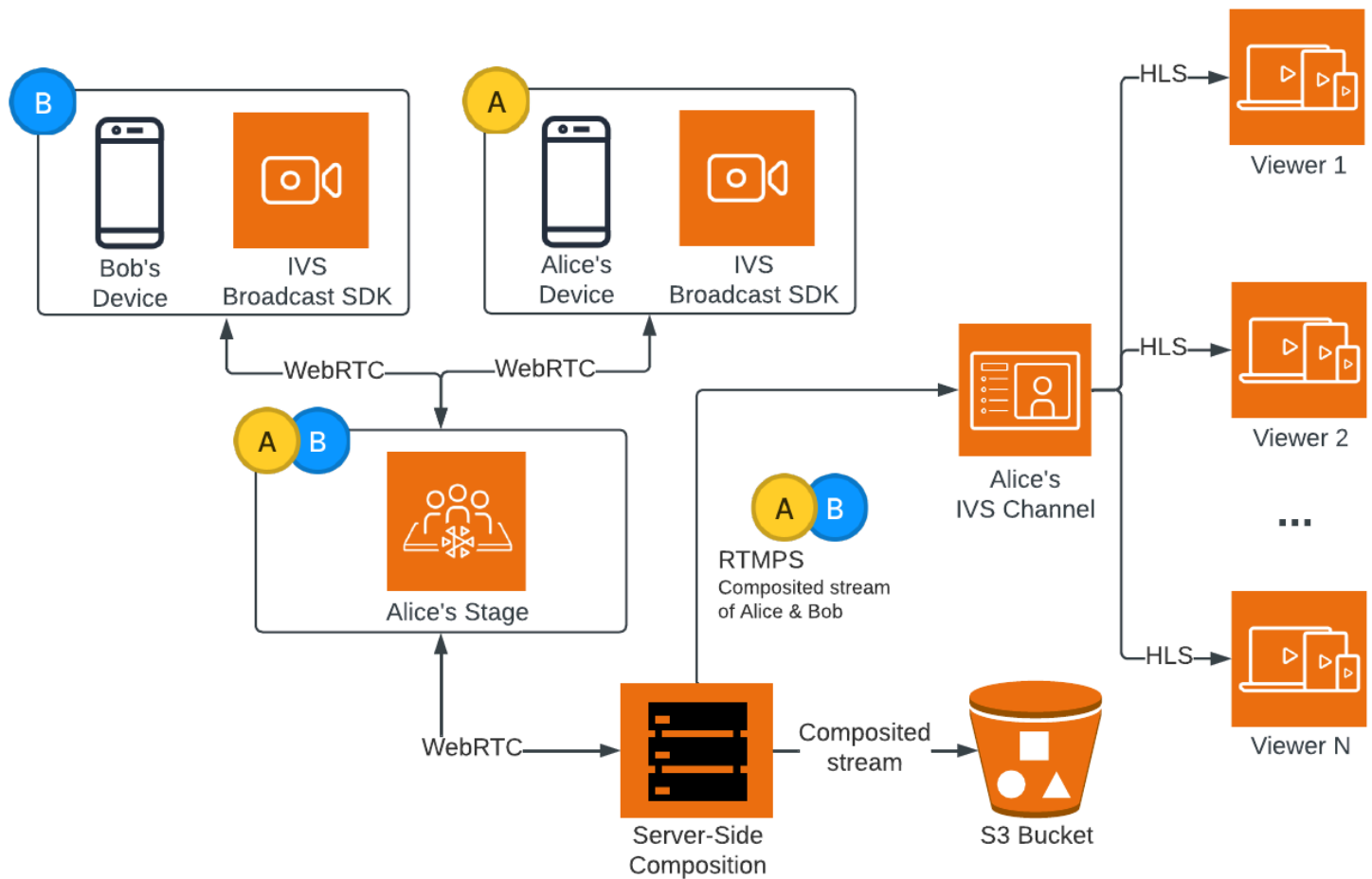
Menyiarkan atau merekam panggung menggunakan komposisi sisi server menawarkan banyak manfaat, menjadikannya pilihan yang menarik bagi pengguna yang mencari alur kerja video berbasis cloud yang efisien dan andal.

Topik

- [Ikhtisar Komposisi Sisi Server IVS](#)
- [Memulai dengan Komposisi Sisi Server IVS](#)
- [Pemesanan Peserta Kustom](#)
- [Mengaktifkan Berbagi Layar dalam Komposisi Sisi Server IVS](#)
- [Masalah dan Solusi yang Diketahui](#)

Ikhtisar Komposisi Sisi Server IVS

Diagram ini menggambarkan cara kerja komposisi sisi server:



Manfaat

Dibandingkan dengan komposisi sisi klien, komposisi sisi server memiliki manfaat sebagai berikut:

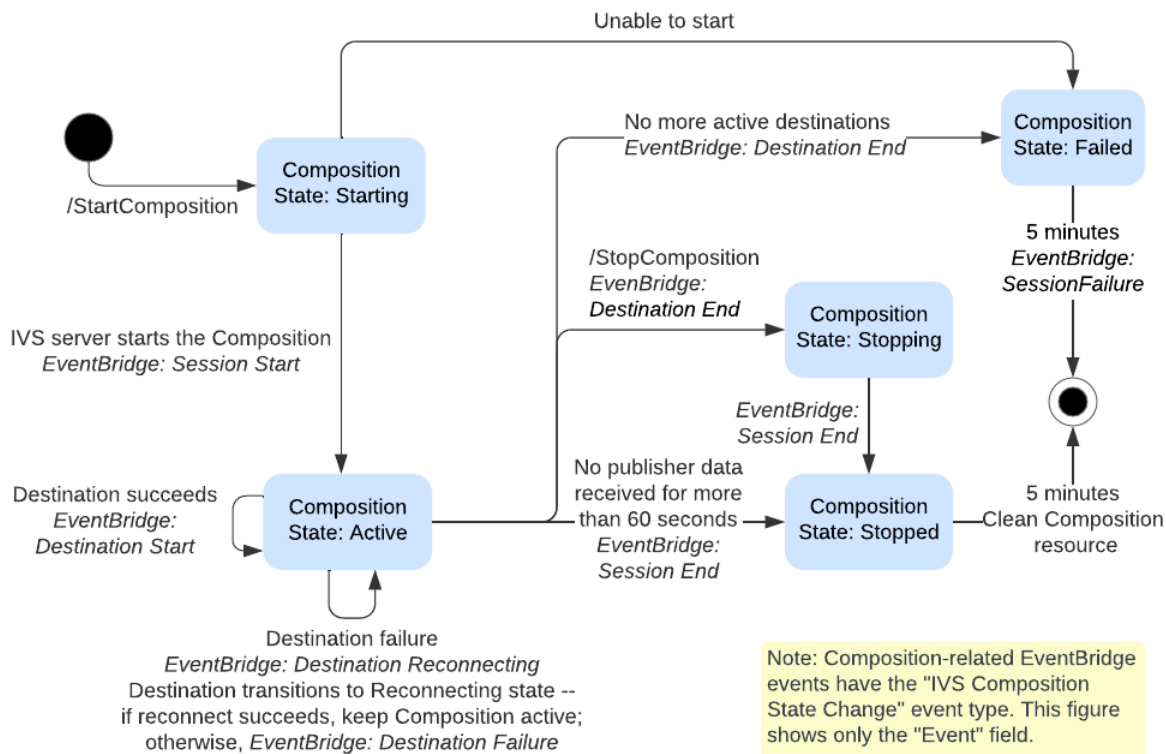
- Mengurangi beban klien - Dengan komposisi sisi server, beban pemrosesan dan penggabungan sumber audio dan video digeser dari perangkat klien individual ke server itu sendiri. Komposisi sisi server menghilangkan kebutuhan perangkat klien untuk menggunakan CPU dan sumber daya jaringan mereka untuk menyusun 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 kepada semua anggota 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](#).

Siklus Hidup Komposisi

Gunakan diagram di bawah ini untuk memahami transisi status komposisi:



Pada tingkat tinggi, siklus hidup suatu Komposisi adalah sebagai berikut:

1. Sumber daya Komposisi dibuat saat pengguna memanggil `StartComposition` operasi.
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` operasi dipanggil, IVS memulai penutupan Komposisi yang anggun, mengirimkan peristiwa "Akhir Tujuan" diikuti dengan acara "Akhir Sesi".

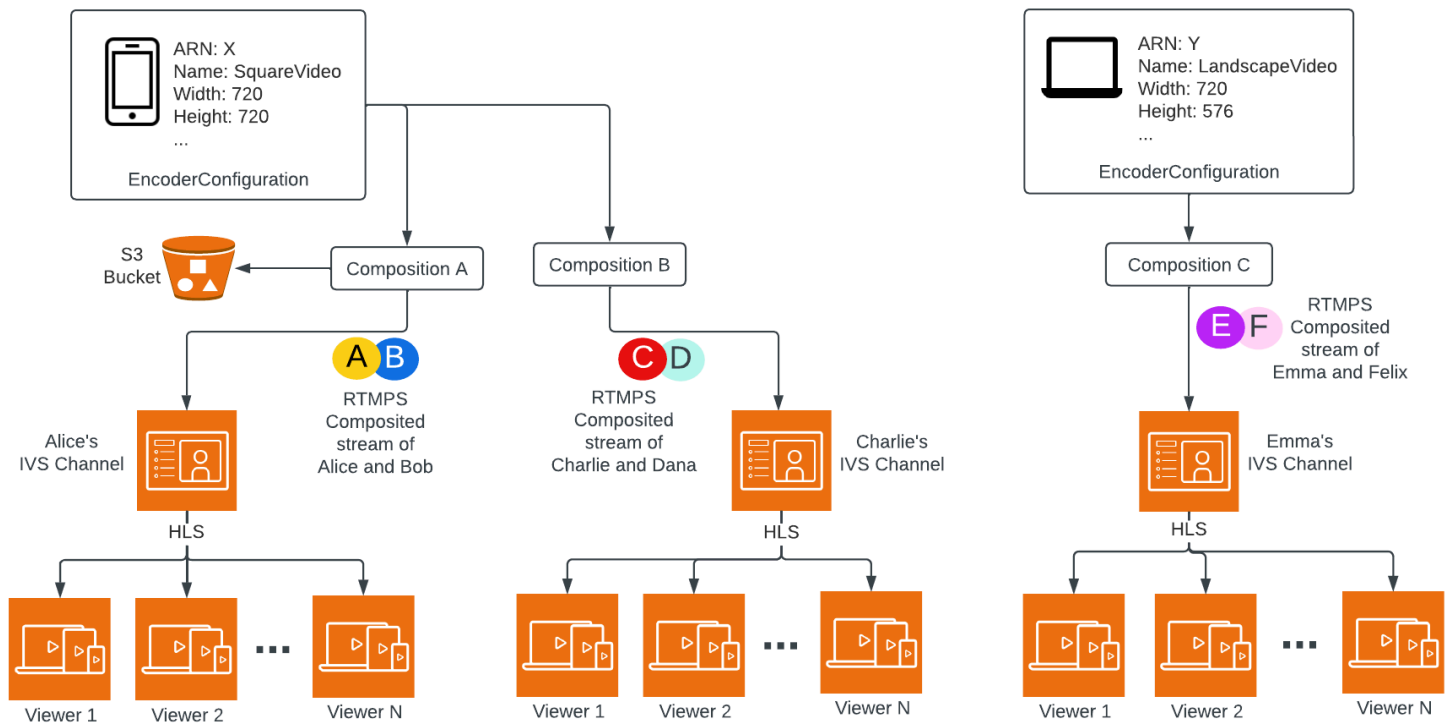
- Komposisi melakukan shutdown otomatis - Jika tahap IVS dihapus atau tidak ada peserta yang secara aktif mempublikasikan ke tahap IVS selama 60 detik, Komposisi diselesaikan secara otomatis dan acara dikirim. EventBridge
 - Kegagalan tujuan — Jika tujuan tiba-tiba gagal (misalnya, saluran IVS akan dihapus), 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.)

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 StartComposition operasi.
- Operasi 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 operasi. StartComposition Dalam contoh ini, SquareVideo EncoderConfiguration digunakan dalam dua Komposisi:



Untuk informasi selengkapnya, lihat [Referensi API Streaming Waktu Nyata IVS](#).

Layout

StartComposition Operasi ini menawarkan dua opsi tata letak: grid dan pip (Picture-in-Picture).

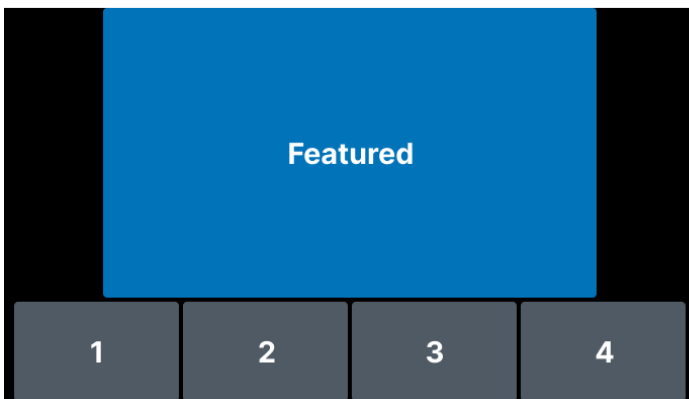
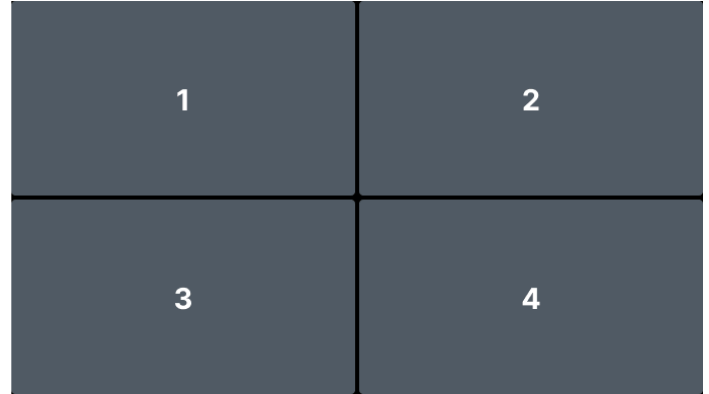
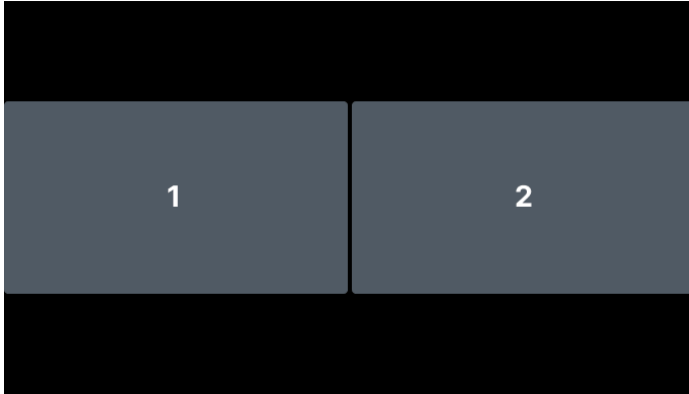
Tata Letak Grid

Tata letak grid mengatur peserta panggung dalam kotak slot berukuran sama. Ini menyediakan beberapa properti yang dapat disesuaikan:

- `videoAspectRatio` mengatur mode tampilan peserta untuk mengontrol rasio aspek ubin video.
- `videoFillMode` mendefinisikan bagaimana konten video cocok dalam ubin peserta.
- `gridGap` menentukan jarak antara ubin peserta dalam piksel.
- `omitStoppedVideo` memungkinkan mengecualikan aliran video yang dihentikan dari komposisi.
- `featuredParticipantAttribute` mengidentifikasi slot unggulan. Ketika ini diatur, peserta unggulan ditampilkan dalam slot yang lebih besar di layar utama, dengan peserta lain ditampilkan di bawahnya.
- `participantOrderAttribute` memungkinkan pemesanan peserta khusus berdasarkan nilai atribut dalam token peserta. Ketika ditentukan, peserta diurutkan secara numerik berdasarkan nilai

atribut mereka, dengan peserta yang tidak memiliki atribut kembali ke urutan waktu kedatangan. Ini memberikan penentuan posisi deterministik opsional dan memungkinkan tata letak berbasis peran.

Untuk detail tentang tata letak kisi (termasuk nilai dan default yang valid untuk semua bidang), lihat tipe data. [GridConfiguration](#)



Picture-in-Picture (PiP) Tata Letak

Tata letak PiP memungkinkan menampilkan peserta di jendela overlay dengan ukuran, posisi, dan perilaku yang dapat dikonfigurasi. Properti utama meliputi:

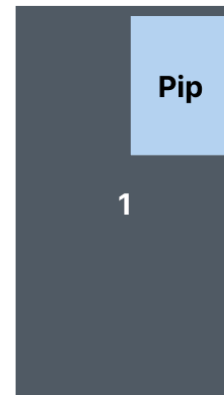
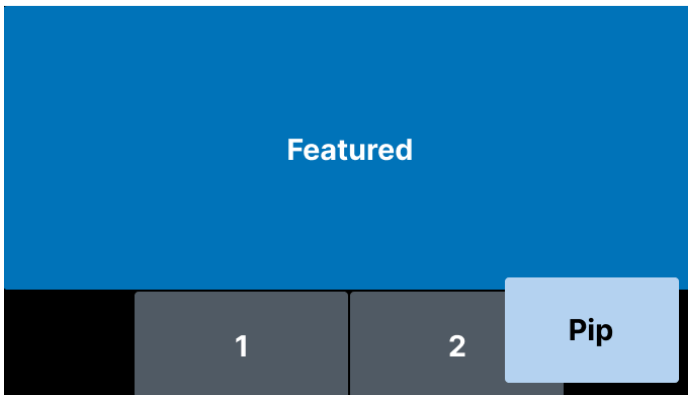
- `pipParticipantAttribute` menentukan peserta untuk jendela PiP.
- `pipPosition` menentukan posisi sudut jendela PiP.
- `pipWidth` dan `pipHeight` mengkonfigurasi lebar dan tinggi jendela PiP.
- `pipOffset` mengatur posisi offset jendela PiP dalam piksel dari tepi terdekat.
- `pipBehavior` mendefinisikan perilaku PiP ketika semua peserta lain telah pergi.

Seperti tata letak kisi, tata letak PiP

mendukung `featuredParticipantAttribute`, `omitStoppedVideo`, `videoFillMode`, `gridGap`, dan `participantOrderAttribute` untuk lebih menyesuaikan komposisi.

`participantOrderAttribute` ini memungkinkan peserta khusus memesan untuk memilih peserta untuk jendela PiP dan memposisikan peserta kisi berdasarkan nilai atribut dalam token peserta.

Untuk detail tentang tata letak PiP (termasuk nilai dan default yang valid untuk semua bidang), lihat tipe data. [PipConfiguration](#)



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.

Penting: Pastikan aplikasi Anda tidak bergantung pada fitur spesifik dari tata letak saat ini, seperti ukuran dan posisi ubin. Perbaikan visual pada tata letak dapat diperkenalkan kapan saja.

Memulai dengan Komposisi Sisi Server IVS

Dokumen ini membawa Anda melalui langkah-langkah yang terlibat dalam memulai dengan komposisi sisi server IVS.

Prasyarat

Untuk menggunakan komposisi sisi server, Anda harus memiliki panggung dengan penerbit aktif dan menggunakan saluran and/or IVS bucket S3 sebagai tujuan komposisi.

Untuk membuat bucket S3, lihat dokumentasi S3 tentang [cara membuat](#) bucket. Bucket S3 harus dibuat di wilayah AWS yang sama dengan tahap IVS.

Penting: Jika Anda menggunakan bucket S3 yang ada:

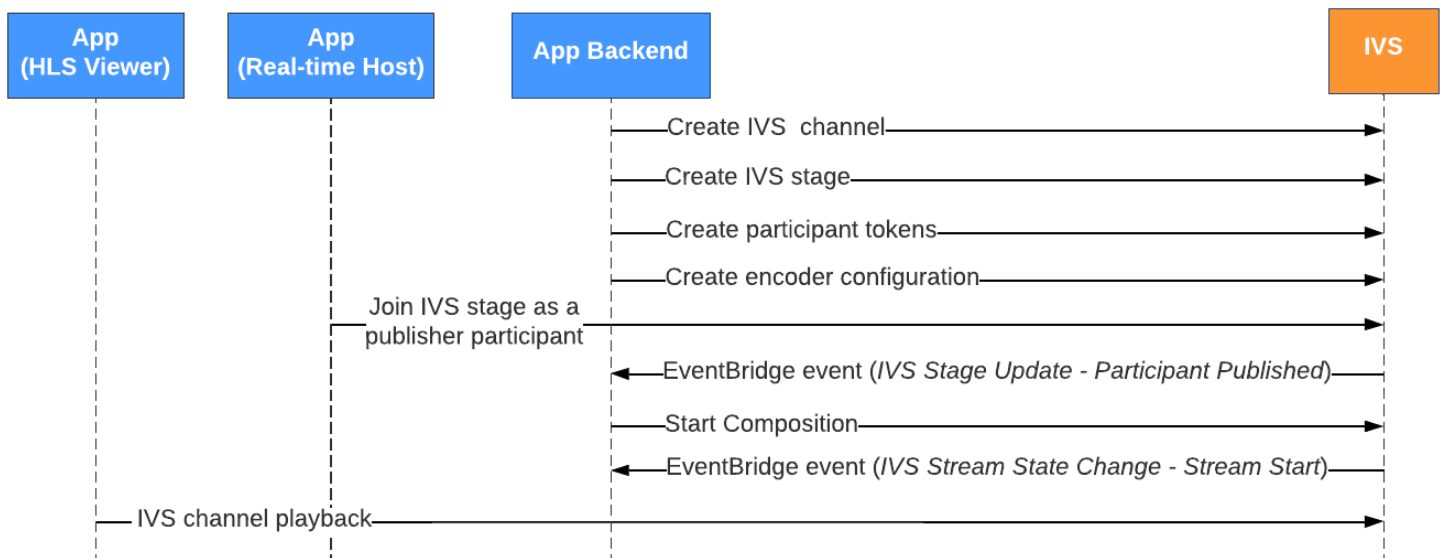
- Pengaturan Kepemilikan Objek harus diberlakukan oleh pemilik Bucket atau pemilik Bucket lebih disukai.
- Pengaturan Enkripsi Default harus berupa enkripsi sisi server dengan kunci terkelola Amazon S3 (SSE-S3).

Untuk detailnya, lihat dokumentasi S3 tentang [mengendalikan kepemilikan objek](#) dan [melindungi data dengan enkripsi](#).

Instruksi API

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 Diterbitkan, hubungi [StartComposition](#) dengan konfigurasi tata letak yang Anda inginkan.
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 detail, lihat [Panduan Pengguna AWS Command Line Interface](#).

Sekarang Anda dapat menggunakan CLI untuk membuat dan mengelola sumber daya. Operasi 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 operasi 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:

Contoh Tata Letak Grid

```
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/D0lMW4dfMR8r", "encoderConfigurationArn": "arn:aws:ivs:us-east-1:927810967299:encoder-configuration/9W590BY2M8s4"}}]' --layout '{"grid": {"participantOrderAttribute": "order", "featuredParticipantAttribute": "isFeatured", "videoFillMode": "fill"}}'
```

Contoh Tata Letak PiP

```
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/D0lMW4dfMR8r", "encoderConfigurationArn": "arn:aws:ivs:us-east-1:927810967299:encoder-configuration/DEkQHWPVa0w0"}}]' --layout '{"pip": {"participantOrderAttribute": "priority", "pipParticipantAttribute": "isPip", "pipOffset": 10, "pipPosition": "bottom-right"}}'
```

Catatan: Anda dapat menggunakan [alat ini](#) untuk lebih mudah menghasilkan --layout konfigurasi berdasarkan pilihan tata letak Anda.

Tanggapan akan menunjukkan bahwa Komposisi dibuat dengan STARTING status. Setelah Komposisi mulai menerbitkan komposisi, status bertransisi keACTIVE. (Anda dapat melihat status dengan memanggil ListCompositions atau GetComposition operasi.)

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.

Pemesanan Peserta Kustom

Pemesanan peserta khusus memungkinkan Anda untuk mengontrol posisi peserta di tata letak kisi dan PiP berdasarkan nilai atribut khusus dalam token peserta, termasuk posisi peserta unggulan

dan pemilihan peserta di jendela PiP. Ini memberikan penentuan posisi peserta deterministik dan memungkinkan tata letak berbasis peran.

Cara Kerja Pemesanan Kustom

Bila `participantOrderAttribute` ditentukan dalam konfigurasi tata letak Anda, peserta diurutkan sesuai dengan aturan berikut:

- Peserta dengan atribut pemesanan yang ditentukan dalam token mereka diposisikan terlebih dahulu, diurutkan secara numerik berdasarkan nilai atribut mereka.
- Peserta tanpa atribut pemesanan kembali ke pemesanan waktu kedatangan dan diposisikan setelah peserta yang dipesan.
- Ketika beberapa peserta memiliki nilai urutan yang identik, mereka diurutkan berdasarkan waktu kedatangan mereka ke panggung.
- Urutan menggunakan pengurutan numerik (bukan leksikografik), jadi "10" muncul setelah "9" (bukan setelah "1").
- Nilai negatif didukung. Mereka diposisikan sebelum nilai-nilai positif.
- Nilai non-numerik (misalnya, "abc", "1.5") diperlakukan sebagai tidak valid, dan peserta tersebut kembali ke pemesanan waktu kedatangan.

Penting: Pemesanan peserta (baik berdasarkan waktu kedatangan atau kebiasaan) berlaku setelah komposisi dimulai. Urutan peserta yang benar tidak dijamin untuk peserta yang bergabung dengan panggung sebelum komposisi dimulai.

Membuat Token dengan Atribut Pemesanan

Untuk menggunakan pemesanan peserta khusus, sertakan atribut pemesanan dalam token peserta Anda saat membuatnya:

```
aws ivs-realtime create-participant-token --stage-arn "arn:aws:ivs:us-east-1:123456789012:stage/u90iE29bT7Xp" --attributes order=1
```

```
aws ivs-realtime create-participant-token --stage-arn "arn:aws:ivs:us-east-1:123456789012:stage/u90iE29bT7Xp" --attributes order=2
```

```
aws ivs-realtime create-participant-token --stage-arn "arn:aws:ivs:us-east-1:123456789012:stage/u90iE29bT7Xp" --attributes order=3
```

Anda dapat menggabungkan custom-participant-order atribut dengan atribut untuk memilih peserta untuk slot unggulan dan jendela PiP:

```
aws ivs-realtime create-participant-token --stage-arn "arn:aws:ivs:us-east-1:123456789012:stage/u90iE29bT7Xp" --attributes order=2,isFeatured=true
```

```
aws ivs-realtime create-participant-token --stage-arn "arn:aws:ivs:us-east-1:123456789012:stage/u90iE29bT7Xp" --attributes order=3,isFeatured=true
```

```
aws ivs-realtime create-participant-token --stage-arn "arn:aws:ivs:us-east-1:123456789012:stage/u90iE29bT7Xp" --attributes order=4,isPip=true
```

Contoh Kasus Penggunaan

Contoh kasus penggunaan meliputi:

- Pemosisian yang konsisten — Peserta mempertahankan posisi mereka saat menyambung kembali dengan token yang sama.
- Posisi berbasis peran — Misalnya, Anda dapat menentukan guru dengan order=1 dan siswa dengan order=2.
- Tata letak berbasis prioritas - Peserta VIP dengan nilai urutan lebih rendah akan muncul terlebih dahulu.
- Tata letak dinamis - Anda dapat menggabungkan pemesanan kustom dengan featuredParticipantAttribute dan pipParticipantAttribute untuk skenario yang kompleks.
- Interaksi lintas tahap — Saat menggunakan replikasi peserta untuk skenario seperti kompetisi Mode VS di mana streamer dari berbagai tahap berinteraksi, Anda dapat mengganti atribut pengurutan untuk mengontrol pemosisian dalam komposisi tahap tujuan.

Catatan: Untuk kasus penggunaan replikasi peserta, Anda dapat mengganti atribut peserta (termasuk atribut urutan) sesuai kebutuhan saat memulai replikasi untuk mencapai tata letak yang diinginkan di tahap tujuan.

Kompatibilitas Ke Belakang

Pemesanan peserta khusus adalah fitur opsional dan sepenuhnya kompatibel ke belakang. Komposisi yang ada tanpa participantOrderAttribute terus bekerja tidak berubah,




```
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/D0lMW4dfMR8r", "encoderConfigurationArn": "arn:aws:ivs:us-east-1:927810967299:encoder-configuration/DEkQHWPVa0w0"}}]' --layout '{"grid":{"featuredParticipantAttribute":"screen-share"}}'
```

Tanggapannya adalah:

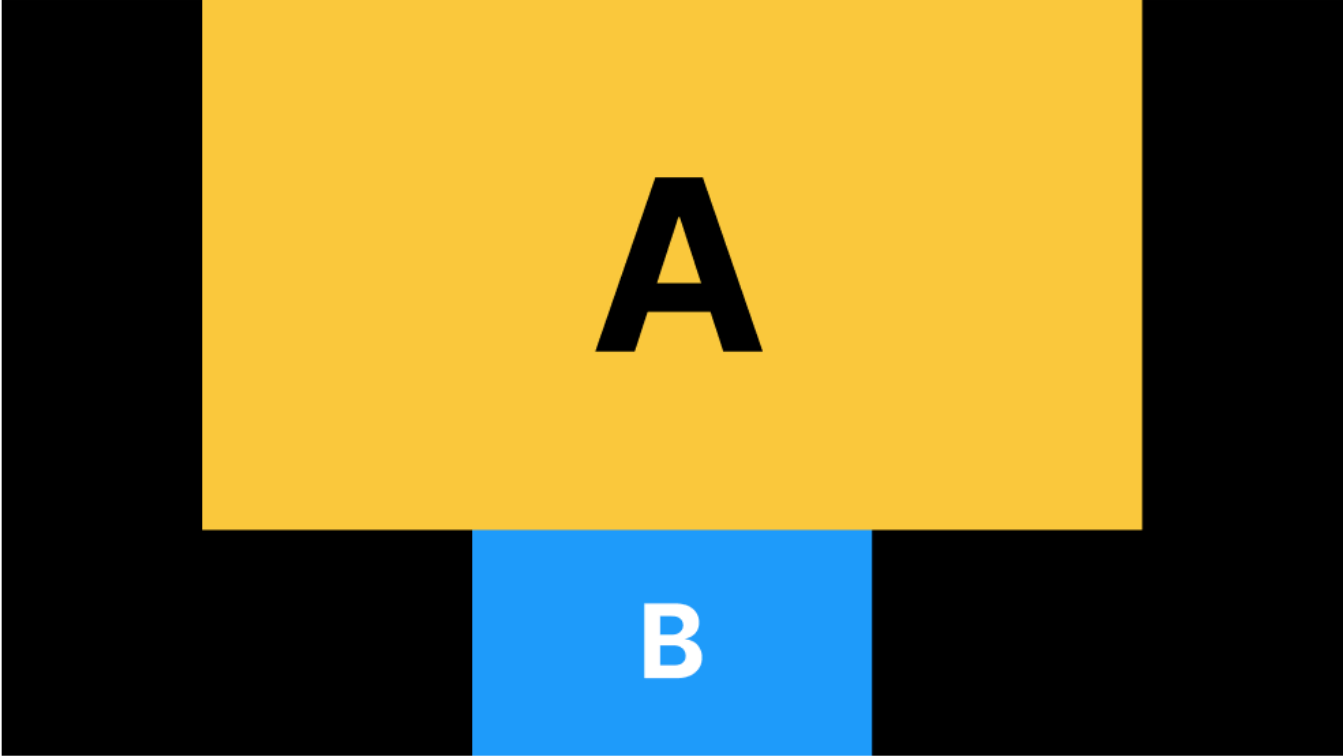
```
{
  "composition" : {
    "arn" : "arn:aws:ivs:us-east-1:927810967299:composition/B19tQcXRgtoz",
    "destinations" : [ {
      "configuration" : {
        "channel" : {
          "channelArn" : "arn:aws:ivs:us-east-1:927810967299:channel/D0lMW4dfMR8r",
          "encoderConfigurationArn" : "arn:aws:ivs:us-east-1:927810967299:encoder-configuration/DEkQHWPVa0w0"
        },
        "name" : ""
      },
      "id" : "SGmgBXTULuXv",
      "state" : "STARTING"
    } ],
    "layout" : {
      "grid" : {
        "featuredParticipantAttribute" : "screen-share",
        "gridGap": 2,
        "omitStoppedVideo": false,
        "videoAspectRatio": "VIDEO"
      }
    },
    "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 kapan saja, hubungi `StopComposition` operasi:

```
aws ivs-realtime stop-composition --arn arn:aws:ivs:us-east-1:927810967299:composition/  
B19tQcXRgtoz
```

Masalah dan Solusi yang Diketahui

Bagian ini mencantumkan masalah yang diketahui yang mungkin Anda temui saat menggunakan komposisi sisi server IVS dan menyarankan solusi potensial.

- Beberapa komposisi mungkin menunjukkan gagap audio singkat setelah periode hening.

Solusi: Tidak ada

Perekaman IVS | Streaming Waktu Nyata

Ada dua opsi perekaman untuk streaming real-time IVS:

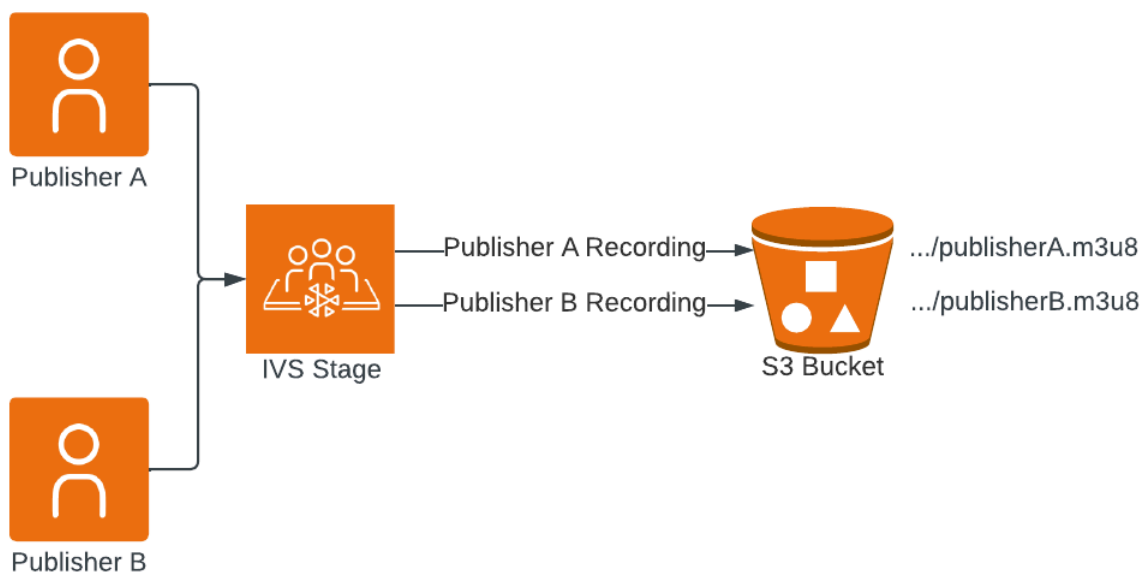
- Dengan rekaman peserta individu, setiap media penerbit direkam dalam file terpisah.
- Sebaliknya, rekaman komposit menggabungkan media dari semua penerbit menjadi satu tampilan dan merekamnya dalam satu file.

Rekaman peserta individu tidak dikenakan biaya Amazon IVS tambahan, sementara rekaman komposit dikenakan biaya untuk tarif per jam untuk video yang dikodekan. Kedua opsi perekaman menimbulkan penyimpanan S3 standar dan biaya permintaan. Untuk detail selengkapnya, lihat [harga Amazon IVS](#).

Untuk solusi yang lebih dapat disesuaikan, pertimbangkan untuk menggunakan proyek [IVSStageSaver](#) sumber terbuka sebagai dasar untuk layanan perekaman yang dihosting sendiri.

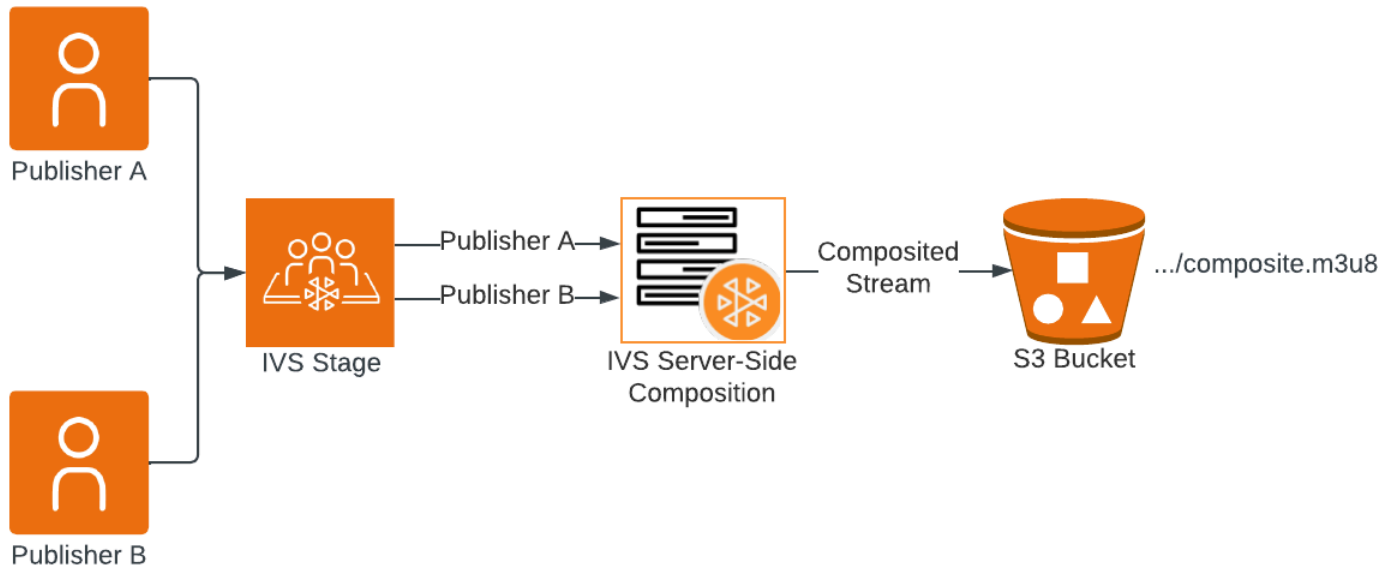
Rekaman Peserta Individu

Opsi ini sangat ideal untuk streaming langsung dengan satu penerbit atau ketika rekaman terpisah dari setiap penerbit diperlukan, terutama untuk tujuan moderasi. Untuk detail selengkapnya, lihat [Rekaman Peserta Individu](#).



Rekaman Komposit

Opsi ini menggabungkan media dari beberapa penerbit menjadi satu tampilan dan mencatatnya dalam satu file, ideal untuk pengalaman video-on-demand. Untuk detail selengkapnya, lihat [Perekaman Komposit](#).



Thumbnail

Rekaman thumbnail untuk streaming real-time IVS dapat diatur untuk rekaman peserta individu dan rekaman komposit (multi-peserta). Untuk mengaktifkan atau menonaktifkan perekaman thumbnail dan menyesuaikan interval di mana thumbnail dihasilkan:

- Untuk rekaman peserta individu, gunakan `thumbnailConfiguration` properti.
- Untuk rekaman komposit, gunakan `thumbnailConfigurations` properti.

Interval thumbnail berkisar dari 1 hingga 86400 detik (24 jam); secara default, perekaman thumbnail dinonaktifkan. Untuk detailnya, lihat [Referensi API Streaming Waktu Nyata Amazon IVS](#).

Konfigurasi thumbnail mencakup `storage` bidang, yang dapat diatur ke `SEQUENTIAL` and/or `LATEST` `storageBidang` menentukan perilaku penyimpanan S3 untuk thumbnail:

- `SEQUENTIAL` menyimpan semua thumbnail secara serial. Ini adalah opsi default.
- `LATEST` hanya menyimpan thumbnail terbaru, menimpa yang sebelumnya.

Jika Anda menentukan keduanya SEQUENTIAL dan LATEST, thumbnail ditulis ke dua jalur S3 terpisah, satu untuk arsip sekuensial dan satu untuk thumbnail terbaru.

Rekaman Peserta Individu IVS | Streaming Waktu Nyata

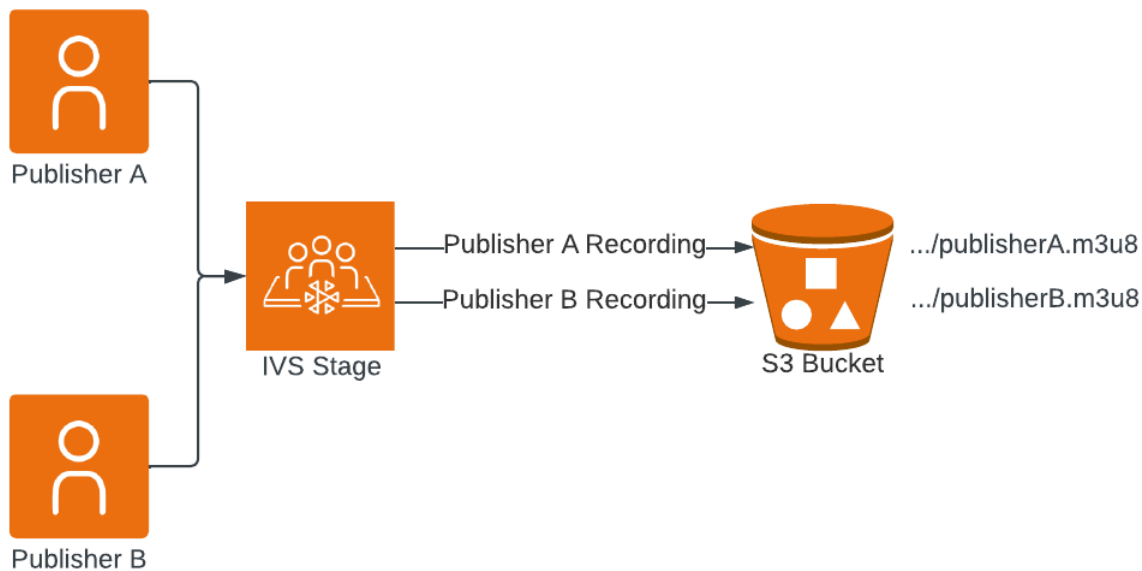
Dokumen ini menjelaskan cara menggunakan rekaman peserta individu dengan streaming real-time IVS.

Biaya penyimpanan dan permintaan S3 standar berlaku. Thumbnail tidak dikenakan biaya IVS tambahan. Untuk detailnya, lihat [Harga Amazon IVS](#).

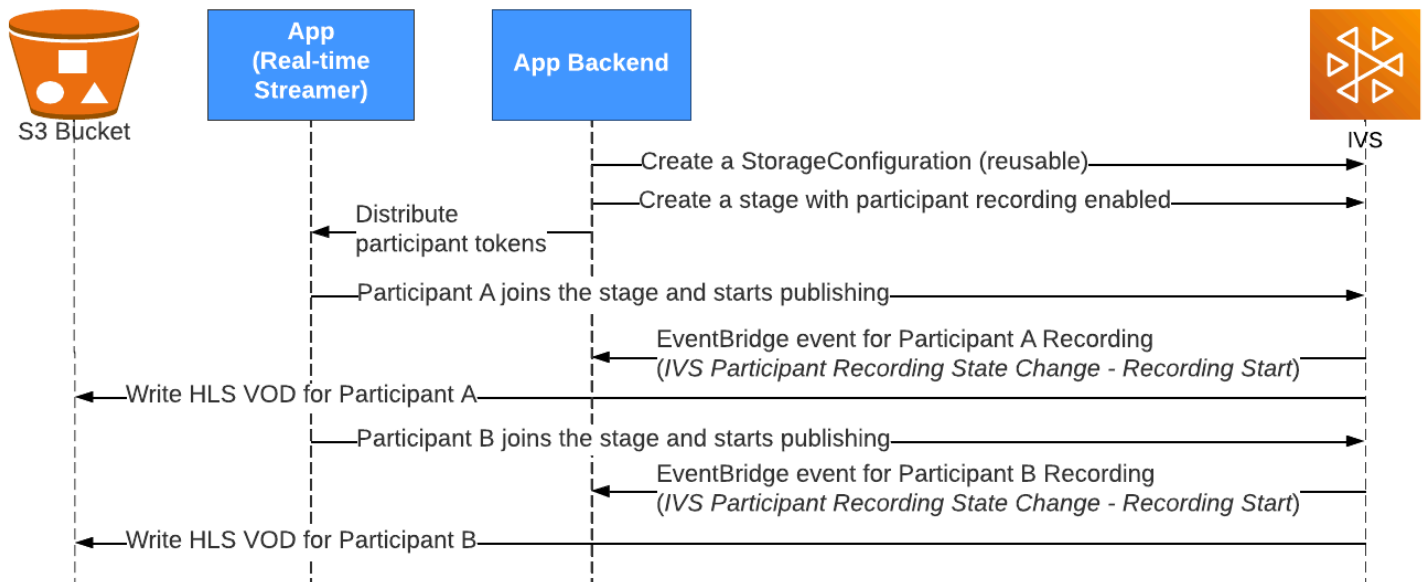
Pengantar

Rekaman peserta individu memungkinkan pelanggan streaming real-time IVS untuk merekam penerbit panggung IVS satu per satu ke dalam ember S3. Ketika rekaman peserta individu diaktifkan untuk sebuah panggung, konten penerbit direkam setelah mereka mulai menerbitkan ke panggung.

Catatan: Jika Anda perlu membuat semua peserta panggung dicampur dalam satu video, fitur perekaman komposit lebih cocok. Lihat [Merekam](#) untuk ringkasan perekaman real-time-streaming konten IVS.



Alur kerja



1. Buat Bucket S3

Anda akan membutuhkan ember S3 untuk menulis VODs. Untuk detailnya, lihat dokumentasi S3 tentang [cara membuat bucket](#). Perhatikan bahwa untuk perekaman peserta individu, bucket S3 harus dibuat di wilayah AWS yang sama dengan tahap IVS.

Penting: Jika Anda menggunakan bucket S3 yang ada:

- Pengaturan Kepemilikan Objek harus diberlakukan oleh pemilik Bucket atau pemilik Bucket lebih disukai.
- Pengaturan Enkripsi Default harus berupa enkripsi sisi server dengan kunci terkelola Amazon S3 (SSE-S3).

Untuk detailnya, lihat dokumentasi S3 tentang [mengendalikan kepemilikan objek](#) dan [melindungi data dengan enkripsi](#).

2. Buat StorageConfiguration Objek

Setelah membuat bucket, panggil API streaming real-time IVS untuk [membuat StorageConfiguration](#) objek. Setelah konfigurasi penyimpanan berhasil dibuat, IVS akan memiliki izin untuk menulis ke bucket S3 yang disediakan. Anda dapat menggunakan kembali StorageConfiguration objek ini pada beberapa tahap.

3. Buat Panggung dengan Token Peserta

Sekarang Anda perlu [membuat tahap IVS](#) dengan rekaman peserta individu diaktifkan (dengan mengatur `AutoParticipantRecordingConfiguration` objek), serta token peserta untuk setiap penerbit.

Permintaan di bawah ini membuat panggung dengan dua token peserta dan rekaman peserta individu diaktifkan.

```
POST /CreateStage HTTP/1.1
Content-type: application/json

{
  "autoParticipantRecordingConfiguration": {
    "mediaTypes": ["AUDIO_VIDEO"],
    "storageConfigurationArn": "arn:aws:ivs:us-west-2:123456789012:storage-configuration/AbCdef1G2hij",
    "thumbnailConfiguration": {
      "recordingMode": "INTERVAL",
      "storage": ["LATEST", "SEQUENTIAL"],
      "targetIntervalSeconds": 60
    }
  },
  "name": "TestStage",
  "participantTokenConfigurations": [
    {
      "capabilities": ["PUBLISH", "SUBSCRIBE"],
      "duration": 20160,
      "userId": "1"
    },
    {
      "capabilities": ["PUBLISH", "SUBSCRIBE"],
      "duration": 20160,
      "userId": "2"
    }
  ]
}
```

4. Bergabunglah dengan Panggung sebagai Penerbit Aktif

Bagikan token peserta ke penerbit Anda, dan minta mereka bergabung dengan panggung dan mulai [mempublikasikannya](#).

Ketika mereka bergabung dengan panggung dan mulai mempublikasikannya menggunakan salah satu [siaran streaming real-time IVS SDKs](#), proses perekaman peserta dimulai secara otomatis dan mengirimkan Anda [EventBridgeacara](#) yang menunjukkan bahwa rekaman dimulai. (Acara ini adalah Perubahan Status Rekaman Peserta IVS - Merekam Mulai.) Secara bersamaan, proses perekaman peserta mulai menulis file VOD dan metadata ke bucket S3 yang dikonfigurasi. Catatan: Peserta yang terhubung untuk jangka waktu yang sangat singkat (kurang dari 5 d) tidak dijamin akan direkam.

Ada dua cara untuk mendapatkan awalan S3 untuk setiap rekaman:

- Dengarkan EventBridge acara:

```
{
  "version": "0",
  "id": "12345678-1a23-4567-a1bc-1a2b34567890",
  "detail-type": "IVS Participant Recording State Change",
  "source": "aws.ivs",
  "account": "123456789012",
  "time": "2024-03-13T22:19:04Z",
  "region": "us-east-1",
  "resources": ["arn:aws:ivs:us-west-2:123456789012:stage/AbCdef1G2hij"],
  "detail": {
    "session_id": "st-ZyXwvu1T2s",
    "event_name": "Recording Start",
    "participant_id": "xYz1c2d3e4f",
    "recording_s3_bucket_name": "ivs-recordings",
    "recording_s3_key_prefix": "<stage_id>/<session_id>/<participant_id>/2024-01-01T12-00-55Z"
  }
}
```

- Gunakan operasi [GetParticipant](#) API — Responsnya mencakup bucket S3 dan awalan tempat peserta direkam. Berikut permintaannya:

```
POST /GetParticipant HTTP/1.1
Content-type: application/json
{
  "participantID": "xYz1c2d3e4f",
  "sessionId": "st-ZyXwvu1T2s",
  "stageArn": "arn:aws:ivs:us-west-2:123456789012:stage/AbCdef1G2hij"
}
```

Dan inilah tanggapannya:

```
Content-type: application/json
{
  "participant": {
    ...
    "recordingS3BucketName": "ivs-recordings",
    "recordingS3Prefix": "<stage_id>/<session_id>/<participant_id>",
    "recordingState": "ACTIVE",
    ...
  }
}
```

5. Mainkan Kembali VOD

Setelah rekaman selesai, Anda dapat menontonnya menggunakan pemutar [IVS](#). Lihat [Pemutaran Konten yang Direkam dari Bucket Pribadi](#) untuk petunjuk tentang pengaturan CloudFront distribusi untuk pemutaran VOD.

Rekaman Hanya Audio

Saat menyiapkan rekaman peserta individu, Anda dapat memilih untuk hanya memiliki segmen HLS audio yang ditulis ke bucket S3 Anda. Untuk menggunakan fitur ini, pilih `AUDIO_ONLY` `mediaType` saat membuat panggung:

```
POST /CreateStage HTTP/1.1
Content-type: application/json

{
  "autoParticipantRecordingConfiguration": {
    "storageConfigurationArn": "arn:aws:ivs:us-west-2:123456789012:storage-configuration/AbCdef1G2hij",
    "mediaTypes": ["AUDIO_ONLY"],
    "thumbnailConfiguration": {
      "recordingMode": "DISABLED"
    }
  },
  "name": "TestStage",
  "participantTokenConfigurations": [
    {
      "capabilities": ["PUBLISH", "SUBSCRIBE"],
      "duration": 20160,

```

```

    "userId": "1"
  },
  {
    "capabilities": ["PUBLISH", "SUBSCRIBE"],
    "duration": 20160,
    "userId": "2"
  }
]
}

```

Rekaman Khusus Thumbnail

Saat menyiapkan rekaman peserta individu, Anda dapat memilih untuk hanya memiliki thumbnail yang ditulis ke bucket S3 Anda. Untuk menggunakan fitur ini, atur `mediaType` ke `NONE` saat membuat panggung. Ini memastikan bahwa tidak ada segmen HLS yang dihasilkan; thumbnail masih dibuat dan ditulis ke bucket S3 Anda.

```

POST /CreateStage HTTP/1.1
Content-type: application/json
{
  "autoParticipantRecordingConfiguration": {
    "storageConfigurationArn": "arn:aws:ivs:us-west-2:123456789012:storage-configuration/AbCdef1G2hij",
    "mediaTypes": ["NONE"],
    "thumbnailConfiguration": {
      "recordingMode": "INTERVAL",
      "storage": ["LATEST", "SEQUENTIAL"],
      "targetIntervalSeconds": 60
    }
  },
  "name": "TestStage",
  "participantTokenConfigurations": [
    {
      "capabilities": ["PUBLISH", "SUBSCRIBE"],
      "duration": 20160,
      "userId": "1"
    },
    {
      "capabilities": ["PUBLISH", "SUBSCRIBE"],
      "duration": 20160,
      "userId": "2"
    }
  ]
}

```

```
}
```

Isi Rekaman

Saat perekaman peserta individu aktif, segmen video HLS, file metadata, dan thumbnail akan mulai ditulis ke bucket S3 yang disediakan saat panggung dibuat. Konten ini tersedia untuk pasca-pemrosesan atau pemutaran sebagai video sesuai permintaan.

Perhatikan bahwa setelah rekaman diselesaikan, acara Perubahan Status Perekaman Peserta IVS - Perekaman Akhir dikirim melalui EventBridge Kami menyarankan Anda memutar kembali atau memproses streaming yang direkam hanya setelah acara ini diterima. Untuk detailnya, lihat [Menggunakan EventBridge dengan IVS Real-Time Streaming](#).

Berikut ini adalah struktur direktori sampel dan isi rekaman sesi IVS langsung:

```
s3://mybucket/stageId/stageSessionId/participantId/timestamp
  events
    recording-started.json
    recording-ended.json
  media
    hls
  multivariant.m3u8
    high
      playlist.m3u8
      1.mp4
  thumbnails
    high
      1.jpg
      2.jpg
  latest_thumbnail
    high
      thumb.jpg
```

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 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 perekaman dan dapat dimainkan dengan pemutar IVS. Jika dikonfigurasi, `latest_thumbnail` subfolder `thumbnails` dan berisi file media thumbnail JPEG yang dihasilkan selama sesi perekaman.

Gabungkan Rekaman Peserta Individu yang Terfragmentasi

`recordingReconnectWindowSeconds` properti pada konfigurasi perekaman memungkinkan Anda menentukan jendela waktu (dalam detik) di mana, jika penerbit panggung terputus dari panggung dan kemudian menyambung kembali, IVS mencoba merekam ke awalan S3 yang sama dengan sesi sebelumnya. Dengan kata lain, jika penerbit terputus dan kemudian menyambung kembali dalam interval yang ditentukan, beberapa rekaman dianggap sebagai rekaman tunggal dan digabungkan bersama.

Jika perekaman thumbnail diaktifkan dalam SEQUENTIAL mode, maka thumbnail juga digabungkan di bawah yang sama. `recordingS3Prefix` Saat rekaman digabungkan, penghitung thumbnail dimulai ulang dari nilai thumbnail sebelumnya yang ditulis untuk rekaman sebelumnya.

Peristiwa Perubahan Status Perekaman IVS di Amazon EventBridge: Perekaman peristiwa Akhir dan file metadata JSON yang berakhir perekaman ditunda setidaknya `recordingReconnectWindowSeconds`, karena IVS menunggu untuk memastikan aliran baru tidak dimulai.

Untuk petunjuk cara menyiapkan fungsionalitas penggabungan aliran, lihat [Langkah 2: Membuat Tahap dengan Perekaman Peserta Opsional](#) dalam Memulai Streaming Waktu Nyata Amazon IVS.

Kelayakan

Agar beberapa rekaman digabungkan menggunakan awalan S3 yang sama, kondisi tertentu harus dipenuhi untuk semua rekaman:

- Nilai `recordingReconnectWindowSeconds` properti `AutoParticipantRecordingConfiguration` untuk tahap ditetapkan lebih besar dari 0.
- Yang `StorageConfigurationArn` digunakan untuk menulis artefak VOD adalah sama untuk setiap rekaman.

- Perbedaan waktu dalam detik antara saat peserta meninggalkan dan bergabung kembali dengan panggung kurang dari atau sama dengan. `recordingReconnectWindowSeconds`

Perhatikan bahwa nilai default `recordingReconnectWindowSeconds` adalah 0, yang menonaktifkan penggabungan.

Sinkronkan Beberapa Rekaman Peserta

Rekaman peserta individu menyertakan `EXT-X-PROGRAM-DATE-TIME` tag dalam daftar putar HLS, yang memberikan stempel waktu UTC yang tepat dengan akurasi milidetik untuk menyinkronkan rekaman dari beberapa peserta selama pasca-pemrosesan.

Saat Anda merekam beberapa peserta satu per satu dan ingin membuat komposisi yang disinkronkan (seperti side-by-side atau picture-in-picture tata letak), Anda dapat menggunakan stempel waktu ini untuk menyelaraskan rekaman secara akurat, bahkan jika peserta bergabung dengan panggung pada waktu yang berbeda atau mengalami diskontinuitas yang berpotensi disebabkan oleh gangguan jaringan.

Daftar putar HLS setiap peserta menyertakan `EXT-X-PROGRAM-DATE-TIME` tag yang menandai:

- Awal rekaman (segmen pertama).
- Setiap titik diskontinuitas selama perekaman; misalnya, ketika jahitan terjadi.

Stempel waktu ini menggunakan presisi milidetik dan disinkronkan di semua peserta menggunakan referensi waktu yang sama.

Contoh Daftar Putar HLS

```
#EXTM3U
#EXT-X-VERSION:7
#EXT-X-TARGETDURATION:12
#EXT-X-PLAYLIST-TYPE:VOD
#EXT-X-MAP:URI="init-0.mp4"
#EXT-X-PROGRAM-DATE-TIME:2024-01-01T12:00:00.000Z
#EXTINF:3.30091,
0.mp4
#EXTINF:5.63794,
1.mp4
#EXTINF:2.74290,
```

```

2.mp4
#EXT-X-DISCONTINUITY
#EXT-X-MAP:URI="init-1.mp4"
#EXT-X-PROGRAM-DATE-TIME:2024-01-01T12:00:52.772Z
#EXTINF:2.54412,
3.mp4
#EXTINF:5.63649,
4.mp4

```

EXT-X-PROGRAM-DATE-TIME Tag memberikan waktu UTC yang tepat untuk segmen pertama dan pada setiap titik diskontinuitas, memungkinkan sinkronisasi yang tepat dengan rekaman peserta lain.

Alur Kerja Sinkronisasi

Untuk menyinkronkan beberapa rekaman peserta, ekstrak EXT-X-PROGRAM-DATE-TIME stempel waktu dari daftar putar HLS masing-masing peserta dan gunakan untuk menghitung offset waktu. Offset ini kemudian dapat diterapkan selama komposisi pasca-pemrosesan menggunakan alat pemrosesan video seperti FFmpeg. Ketika diskontinuitas hadir dalam rekaman, stempel waktu pada titik-titik tersebut memberikan referensi waktu yang diperlukan untuk mempertahankan sinkronisasi yang akurat di seluruh rekaman.

Catatan: Untuk output yang disinkronkan secara real-time tanpa pasca-pemrosesan, pertimbangkan untuk menggunakan komposisi sisi server alih-alih perekaman peserta individu.

File Metadata JSON

Metadata ini dalam format JSON. Ini terdiri dari informasi berikut:

Bidang	Tipe	Diperlukan	Deskripsi
stage_arn	string	Ya	ARN panggung digunakan sebagai sumber rekaman.
session_id	string	Ya	String mewakili panggung session_id di mana peserta direkam.
participant_id	string	Ya	String yang mewakili pengidentifikasi peserta yang direkam.

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_START_FAILED</code> . Juga, lihat catatan di bawah ini untuk <code>recording_ended_at</code> .
<code>recording_ended_at</code>	string	Bersyarat	Stempel waktu RFC 3339 UTC saat rekaman berakhir. Ini hanya tersedia jika <code>recording_status</code> ada <code>"RECORDING_ENDED"</code> atau <code>"RECORDING_ENDED_WITH_FAILURE"</code> . Catatan: <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.
<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> .

Bidang	Tipe	Diperlukan	Deskripsi
<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>media</code>	object	Ya	Objek yang berisi objek yang disebutkan dari konten media yang tersedia untuk rekaman ini. Nilai valid: <code>"hls"</code> .
<code>hls</code>	object	Ya	Bidang yang disebutkan yang menjelaskan output format Apple HLS.
<code>duration_ms</code>	integer	Bersyarat	Durasi konten HLS yang direkam dalam milidetik. Ini hanya tersedia jika <code>recording_status</code> ada <code>"RECORDING_ENDED"</code> atau <code>"RECORDING_ENDED_WITH_FAILURE"</code> . Jika terjadi kegagalan sebelum perekaman dilakukan, ini adalah 0.
<code>path</code>	string	Ya	Jalur relatif dari awalan S3 tempat konten HLS disimpan.
<code>playlist</code>	string	Ya	Nama file daftar putar master HLS.
<code>renditions</code>	object	Ya	Array rendisi (varian HLS) objek metadata. Selalu ada setidaknya satu rendisi.

Bidang	Tipe	Diperlukan	Deskripsi
<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>thumbnails</code>	object	Bersyarat	Bidang enumerasi yang menjelaskan output thumbnail. Ini hanya tersedia jika bidang konfigurasi thumbnail termasuk <code>storage SEQUENTIAL</code>
<code>path</code>	string	Ya	Jalur relatif dari awalan S3 tempat konten thumbnail berurutan disimpan.
<code>renditions</code>	object	Ya	Array rendisi (varian thumbnail) objek metadata. Selalu ada setidaknya satu rendisi.
<code>path</code>	string	Ya	Jalur relatif dari awalan S3 tempat konten thumbnail disimpan untuk rendisi ini.
<code>latest_thumbnail</code>	object	Bersyarat	Bidang enumerasi yang menjelaskan output thumbnail. Ini hanya tersedia jika <code>storage</code> bidang konfigurasi thumbnail disertakan. <code>LATEST</code>
<code>path</code>	string	Ya	Jalur relatif dari awalan S3 tempat <code>latest_thumbnail</code> disimpan.

Bidang	Tipe	Diperlukan	Deskripsi
renditions	object	Ya	Array rendisi (varian thumbnail) objek metadata. Selalu ada setidaknya satu rendisi.
path	string	Ya	Jalur relatif dari awalan S3 tempat thumbnail terbaru disimpan untuk rendisi ini.
version	string	Ya	Versi skema metadata.

Contoh: recording-started.json

```
{
  "version": "v1",
  "stage_arn": "arn:aws:ivs:us-west-2:aws_account_id:stage/AbCdef1G2hij",
  "session_id": "st-ZyXwvu1T2s",
  "participant_id": "xYz1c2d3e4f",
  "recording_started_at": "2024-03-13T13:17:17Z",
  "recording_status": "RECORDING_STARTED",
  "media": {
    "hls": {
      "path": "media/hls",
      "playlist": "multivariant.m3u8",
      "renditions": [
        {
          "path": "high",
          "playlist": "playlist.m3u8"
        }
      ]
    },
    "thumbnails": {
      "path": "media/thumbnails",
      "renditions": [
        {
          "path": "high"
        }
      ]
    }
  },
}
```

```

    "latest_thumbnail": {
      "path": "media/latest_thumbnail",
      "renditions": [
        {
          "path": "high"
        }
      ]
    }
  }
}

```

Contoh: recording-ended.json

```

{
  "version": "v1",
  "stage_arn": "arn:aws:ivs:us-west-2:aws_account_id:stage/AbCdef1G2hij",
  "session_id": "st-ZyXwvu1T2s",
  "participant_id": "xYz1c2d3e4f",
  "recording_started_at": "2024-03-13T19:44:19Z",
  "recording_ended_at": "2024-03-13T19:55:04Z",
  "recording_status": "RECORDING_ENDED",
  "media": {
    "hls": {
      "duration_ms": 645237,
      "path": "media/hls",
      "playlist": "multivariant.m3u8",
      "renditions": [
        {
          "path": "high",
          "playlist": "playlist.m3u8"
        }
      ]
    },
    "thumbnails": {
      "path": "media/thumbnails",
      "renditions": [
        {
          "path": "high"
        }
      ]
    },
    "latest_thumbnail": {
      "path": "media/latest_thumbnail",

```

```

    "renditions": [
      {
        "path": "high"
      }
    ]
  }
}

```

Contoh: perekaman-failed.json

```

{
  "version": "v1",
  "stage_arn": "arn:aws:ivs:us-west-2:aws_account_id:stage/AbCdef1G2hij",
  "session_id": "st-ZyXwvu1T2s",
  "participant_id": "xYz1c2d3e4f",
  "recording_started_at": "2024-03-13T19:44:19Z",
  "recording_ended_at": "2024-03-13T19:55:04Z",
  "recording_status": "RECORDING_ENDED_WITH_FAILURE",
  "media": {
    "hls": {
      "duration_ms": 645237,
      "path": "media/hls",
      "playlist": "multivariant.m3u8",
      "renditions": [
        {
          "path": "high",
          "playlist": "playlist.m3u8"
        }
      ]
    }
  },
  "thumbnails": {
    "path": "media/thumbnails",
    "renditions": [
      {
        "path": "high"
      }
    ]
  },
  "latest_thumbnail": {
    "path": "media/latest_thumbnail",
    "renditions": [
      {

```

```
    "path": "high"
  }
]
}
}
```

Mengonversi Rekaman ke MP4

Rekaman peserta individu disimpan dalam format HLS, yang terdiri dari daftar putar dan segmen terfragmentasi MP4 (fMP4). Untuk mengonversi rekaman HLS menjadi satu MP4 file, instal FFmpeg dan jalankan perintah berikut:

```
ffmpeg -i /path/to/playlist.m3u8 -i /path/to/playlist.m3u8 -map 0:v -map 1:a -c copy
output.mp4
```

Rekaman Komposit IVS | 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.

Biaya penyimpanan dan permintaan S3 standar berlaku. Thumbnail tidak dikenakan biaya IVS tambahan. Untuk detailnya, lihat [Harga Amazon IVS](#).

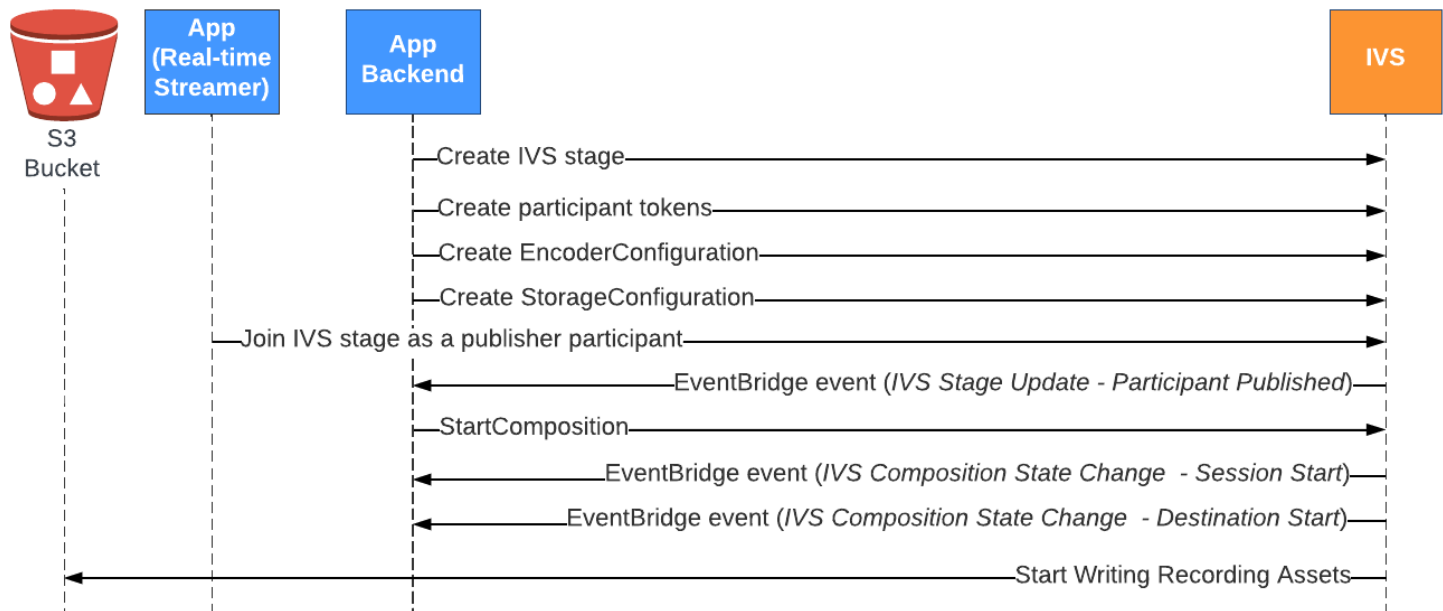
Prasyarat

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).

Penting: Jika Anda menggunakan bucket S3 yang sudah ada, setelah Kepemilikan Objek harus diberlakukan oleh pemilik Bucket atau lebih disukai pemilik Bucket. Untuk detailnya, lihat dokumentasi S3 tentang [pengendalian kepemilikan objek](#).

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](#) operasi, 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/ZBcEbgBE24Cq",
        "thumbnailConfigurations": [
          {
            "storage": ["LATEST", "SEQUENTIAL"],
            "targetIntervalSeconds": 30
          }
        ]
      }
    }
  ],
  "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/ZBcEbgBE24Cq",
      "thumbnailConfigurations": [
        {
          "storage": ["LATEST", "SEQUENTIAL"],
          "targetIntervalSeconds": 30
        }
      ]
    },
    "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, segmen video HLS, file metadata, dan thumbnail (jika dikonfigurasi) akan mulai ditulis ke bucket S3 yang ditentukan selama panggilan berlangsung. `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, segmen video, dan thumbnail

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
    thumbnails
    latest_thumbnail
```

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. Jika dikonfigurasi, latest_thumbnail subfolder thumbnails dan berisi file media thumbnail JPEG yang dihasilkan selama sesi komposisi.

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:

JSON

```
{
  "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	object	Ya	Objek yang berisi objek yang disebutkan dari konten media yang tersedia untuk rekaman ini. Nilai yang valid:"hls".
hls	object	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 recording_status 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.
renditions	object	Ya	Array rendisi (varian HLS) objek metadata. Selalu ada setidaknya satu rendisi.
path	string	Ya	Jalur relatif dari awalan S3 tempat konten HLS disimpan untuk rendisi ini.

Bidang	Tipe	Diperlukan	Deskripsi
<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>thumbnails</code>	object	Bersyarat	Bidang enumerasi yang menjelaskan output thumbnail. Ini hanya tersedia jika bidang konfigurasi thumbnail termasuk <code>storage SEQUENTIAL</code>
<code>path</code>	string	Ya	Jalur relatif dari awalan S3 tempat konten thumbnail berurutan disimpan.
<code>resolutions</code>	object	Ya	Array resolusi (varian thumbnail) objek metadata. Setidaknya selalu ada satu resolusi.
<code>path</code>	string	Ya	Jalur relatif dari awalan S3 tempat konten thumbnail disimpan untuk resolusi ini.
<code>resolution_height</code>	int	Ya	Tinggi resolusi piksel dari thumbnail.
<code>resolution_width</code>	int	Ya	Lebar resolusi piksel dari thumbnail.

Bidang	Tipe	Diperlukan	Deskripsi
<code>latest_thumbnail</code>	object	Bersyarat	Bidang enumerasi yang menjelaskan output thumbnail. Ini hanya tersedia jika <code>storage</code> bidang konfigurasi thumbnail disertakan. LATEST
<code>path</code>	string	Ya	Jalur relatif dari awalan S3 tempat <code>latest_thumbnail</code> disimpan.
<code>resolutions</code>	object	Ya	Array resolusi (varian thumbnail) objek metadata. Setidaknya selalu ada satu resolusi.
<code>path</code>	string	Ya	Jalur relatif dari awalan S3 tempat thumbnail terbaru disimpan untuk resolusi ini.
<code>resolution_height</code>	int	Ya	Tinggi resolusi piksel dari thumbnail terbaru.
<code>resolution_width</code>	int	Ya	Lebar resolusi piksel dari thumbnail terbaru.

Bidang	Tipe	Diperlukan	Deskripsi
<code>recording_ended_at</code>	string	Bersyarat	<p>Stempel waktu RFC 3339 UTC saat rekaman berakhir. Ini hanya tersedia jika <code>recording_status</code> ada <code>"RECORDING_ENDED"</code> atau <code>"RECORDING_ENDED_WITH_FAILURE"</code>.</p> <p><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.</p>
<code>recording_started_at</code>	string	Bersyarat	<p>Stempel waktu RFC 3339 UTC saat perekaman dimulai. Ini tidak tersedia kapan <code>recording_status</code> <code>RECORDING_START_FAILED</code>.</p> <p>Lihat catatan di atas untuk <code>recording_ended_at</code>.</p>
<code>recording_status</code>	string	Ya	<p>Status rekaman. Nilai yang valid: <code>"RECORDING_STARTED"</code>, <code>"RECORDING_ENDED"</code>, <code>"RECORDING_START_FAILED"</code>, <code>"RECORDING_ENDED_WITH_FAILURE"</code>.</p>

Bidang	Tipe	Diperlukan	Deskripsi
recording_status_message	string	Bersyarat	Informasi deskriptif tentang status. Ini hanya tersedia jika recording_status ada "RECORDING_ENDED" atau "RECORDING_ENDED_WITH_FAILURE".
version	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
        }
      ]
    }
  },
  "thumbnails": {
    "path": "media/thumbnails",
    "resolutions": [
      {
        "path": "1280x720",
        "resolution_width": 1280,
        "resolution_height": 720
      }
    ]
  }
},
```

```
"latest_thumbnail": {
  "path": "media/latest_thumbnail",
  "resolutions": [
    {
      "path": "1280x720",
      "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
        }
      ]
    }
  },
  "thumbnails": {
    "path": "media/thumbnails",
    "resolutions": [
      {
        "path": "1280x720",
        "resolution_width": 1280,
        "resolution_height": 720
      }
    ]
  }
}
```

```

    ]
  },
  "latest_thumbnail": {
    "path": "media/latest_thumbnail",
    "resolutions": [
      {
        "path": "1280x720",
        "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
        }
      ]
    }
  },
  "thumbnails": {
    "path": "media/thumbnails",
    "resolutions": [
      {
        "path": "1280x720",
        "resolution_width": 1280,

```

```
        "resolution_height": 720
      }
    ]
  },
  "latest_thumbnail": {
    "path": "media/latest_thumbnail",
    "resolutions": [
      {
        "path": "1280x720",
        "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 IDs yang relevan dengan CloudFront distribusi Anda dan StorageConfiguration.

JSON

```

{
  "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"
        }
      }
    }
  ]
}

```

Pemecahan 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 atas panggung selama lebih dari 60 detik.

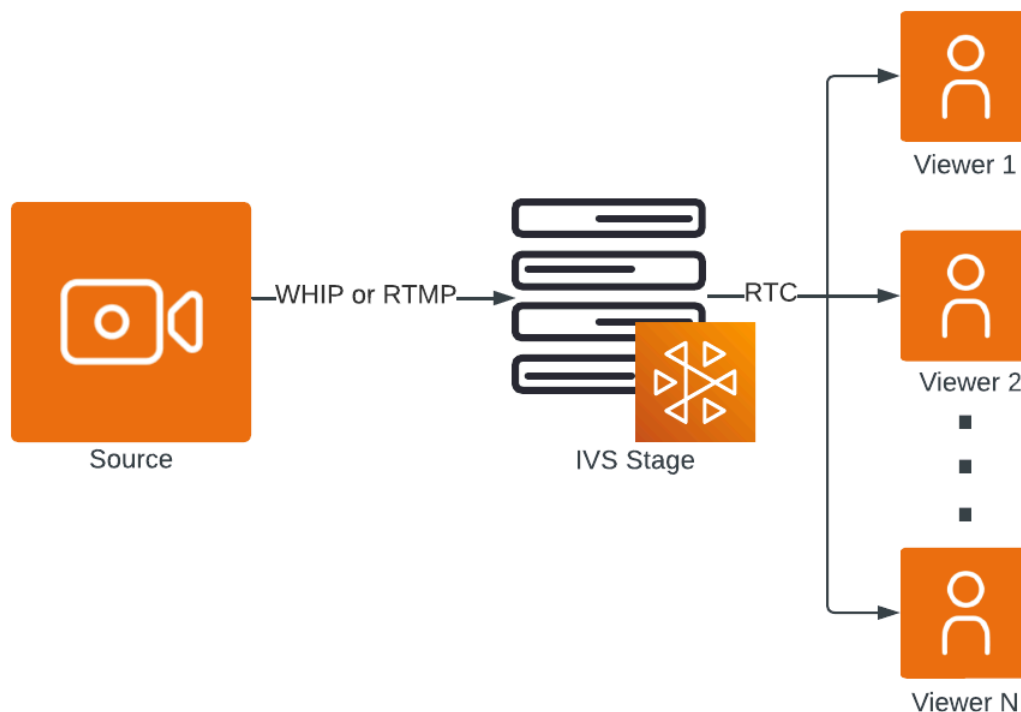
Masalah yang Diketahui

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.

IVS Stream Ingest | Streaming Waktu Nyata

Sebagai alternatif untuk menggunakan SDK siaran IVS, Anda dapat mempublikasikan video ke tahap IVS dari sumber WHIP atau RTMP. Pendekatan ini menawarkan fleksibilitas untuk alur kerja di mana penggunaan SDK tidak layak atau disukai, seperti saat menerbitkan video dari OBS Studio atau encoder perangkat keras. Jika memungkinkan, kami sarankan untuk menggunakan SDK siaran IVS, karena kami tidak dapat menjamin kinerja atau kompatibilitas solusi pihak ketiga dengan IVS.

Diagram ini menggambarkan bagaimana penerbitan dengan WHIP dan RTMP bekerja:



Protokol yang Didukung

Streaming real-time IVS mendukung beberapa protokol ingest:

- RTMP dan RTMPS — RTMP (Real-Time Messaging Protocol) adalah standar industri untuk mentransmisikan video melalui jaringan. RTMPS adalah versi aman dari RTMP yang beroperasi melalui TLS.

IVS mendukung kemampuan video multitrack E-RTMP (Enhanced RTMP). Lihat [E-RTMP Multitrack Video](#) dalam dokumentasi IVS RTMP Publishing.

- WHIP (WebRTC-HTTP Ingestion Protocol) — Draf IETF yang dikembangkan untuk membakukan konsumsi WebRTC.

[Untuk panduan rinci tentang penggunaan protokol ini, lihat dokumentasi RTMP dan WHIP kami.](#)

Spesifikasi Media yang Didukung

- Format masukan audio
 - Codec: AAC-LC untuk RTMP dan Opus untuk WHIP
 - Saluran: 2 (Stereo) atau 1 (Mono)
 - Tingkat sampel: 44,1 kHz atau 48 kHz
 - Bitrate maksimum: 160 Kbps
- Format masukan video
 - Codec: H.264
 - Profil H.264: Baseline
 - Interval IDR: 1 atau 2 detik
 - Frame rate: 10 hingga 60 FPS
 - Bingkai-B: 0

Catatan: SDK siaran IVS memiliki B-frame yang diaktifkan secara default, tetapi dimulai dengan versi 1.25.0, secara otomatis menonaktifkan bingkai-B saat menyiarkan ke tahap IVS. Untuk streaming real-time dengan encoder RTMP lainnya, pengembang harus menonaktifkan B-frame. Jika pengembang yang menggunakan encoder RTMP lain tidak menonaktifkan B-frame, aliran mereka akan terputus.

- Resolusi: Maksimum: 720p. Minimal: 160p
- Bitrate maksimum: 8,5 Mbps

Catatan: Untuk stream RTMP single-track, batas ini berlaku untuk trek tersebut. Untuk video multitrack yang diterbitkan menggunakan Enhanced RTMP, batasnya berlaku untuk bitrate gabungan dari semua trek video.

- Konfigurasi encoder: Kami merekomendasikan penggunaan `veryfast` dan `zerolatency` pengaturan untuk encoder H.264. Juga: opsi `sliced_threads x264` termasuk dalam `zerolatency` preset, dan kami sarankan Anda menonaktifkannya. Misalnya, saat

menggunakan FFmpeg, perintah Anda harus mencakup: `-preset:v veryfast -tune zerolatency -x264-params sliced-threads=0`

Penerbitan IVS RTMP | Streaming Waktu Nyata

Dokumen ini menguraikan proses penerbitan ke tahap IVS menggunakan RTMP. Untuk detail tambahan tentang berbagai opsi konsumsi, lihat dokumentasi [Stream Ingest](#)

Prasyarat

Buat Panggung

Untuk membuat panggung, gunakan perintah berikut:

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

Lihat [CreateStage](#) detailnya, termasuk tanggapannya.

Penting: Dalam tanggapan, perhatikan endpoints bidang, yang mencantumkan titik akhir RTMP dan RTMPS. Ini diperlukan untuk mengatur encoder RTMP Anda.

Buat Konfigurasi Ingest

Untuk mempublikasikan ke panggung menggunakan RTMPS, Anda harus terlebih dahulu membuat konfigurasi ingest dan mengaitkannya dengan panggung Anda. Saat Anda mempublikasikan ke panggung (menggunakan kunci aliran dari konfigurasi ingest dan titik akhir RTMP dari panggung), media akan dipublikasikan ke panggung sebagai peserta. Anda memiliki opsi untuk menentukan `userId` dan `kustomattributes`, yang akan dikaitkan dengan [peserta](#) yang terhubung ke panggung.

```
aws ivs-realtime create-ingest-configuration \  
  --name 'test' \  
  --stage-arn arn:aws:ivs:us-east-1:123456789012:stage/8faHz1SQp0ik \  
  --user-id '123' \  
  --ingest-protocol 'RTMPS'
```

Lihat [CreateIngestConfiguration](#) detailnya, termasuk tanggapannya.

Saat membuat konfigurasi ingest, Anda dapat mengaitkannya dengan ARN tahap tertentu di depan. Tanpa asosiasi ini, kunci aliran tidak dapat digunakan. Juga, konfigurasi ingest (termasuk `stageArn`

bidang) dapat diperbarui melalui [UpdateIngestConfiguration](#) operasi, memungkinkan Anda untuk menggunakan kembali konfigurasi yang sama untuk tahapan yang berbeda.

Catatan: `insecureIngest` Bidang konfigurasi ingest default ke `false`, membutuhkan penggunaan RTMPS. Koneksi RTMP akan ditolak. Jika Anda harus menggunakan RTMP, atur `insecureIngest` ke `true`. Sebaiknya gunakan RTMPS kecuali Anda memiliki kasus penggunaan spesifik dan terverifikasi yang memerlukan RTMP.

Video Trek Tunggal RTMP

[Di bawah ini kami menjelaskan cara menggunakan OBS Studio; Namun, Anda dapat menggunakan encoder RTMP apa pun yang memenuhi spesifikasi media IVS.](#)

Panduan OBS

1. Unduh dan instal perangkat lunak: <https://obsproject.com/unduh>.
2. Klik Pengaturan. Di bagian Stream pada panel Pengaturan, pilih Kustom dari menu tarik-turun Layanan.
3. Untuk Server, masukkan titik akhir RTMP atau RTMPS dari panggung.
4. Untuk Stream Key, masukkan `streamKey` dari konfigurasi ingest.
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 sangat cepat 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.
 - d. Nonaktifkan B-frame, karena aliran dengan B-frame akan terputus secara otomatis. Lakukan salah satu tindakan berikut:
 - Dalam opsi `x264`, masukkan. `bframes=0 sliced-threads=0`
 - Setel B-frame ke 0 jika itu adalah opsi (misalnya, untuk `NVENC`).

Catatan: Streaming RTMP harus menyertakan trek audio dan video, atau mereka akan terputus.

6. Pilih Mulai Streaming

Penting: Jika bitrate maksimum encoder Anda disetel ke 8,5 Mbps, penerbit terkadang menghilang dari sesi. Ini karena pengaturan bitrate maksimum hanya target, dan encoder kadang-kadang melampaui target. Untuk mencegah hal ini, atur bitrate maksimum encoder Anda lebih rendah; misalnya ke 6 Mbps.

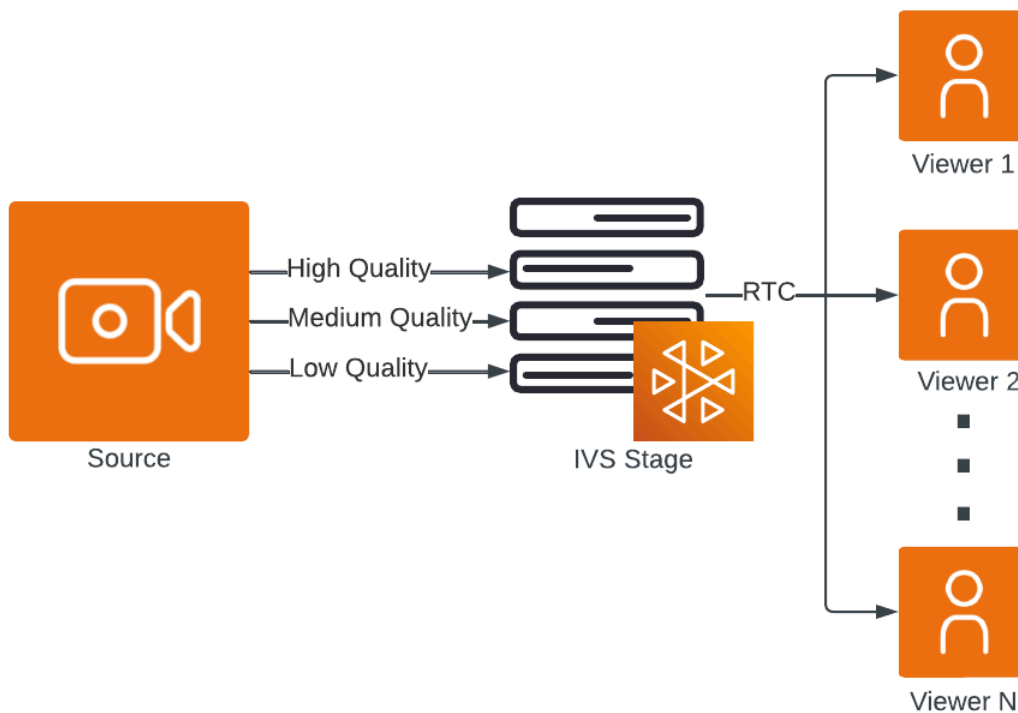
E-RTMP Multitrack Video

IVS mendukung kemampuan video multitrack E-RTMP (Enhanced Real-Time Messaging Protocol), yang memungkinkan Anda mempublikasikan beberapa kualitas video dalam satu aliran RTMP ke tahap IVS Anda. Ini memungkinkan streaming bitrate adaptif, sehingga pelanggan dapat secara otomatis menonton dengan kualitas terbaik untuk koneksi jaringan mereka.

Setelah dicerna, kualitas video yang berbeda dikirimkan ke pelanggan sebagai lapisan simulcast. [Untuk mengonfigurasi lapisan mana yang diterima oleh pelanggan, lihat bagian “Layered Encoding with Simulcast” di panduan SDK siaran streaming real-time: Android, iOS, dan Web.](#)

Untuk kode sampel, lihat [sample-amazon-ivs-multitrackaws-samples/](#) -video di. GitHub

Diagram ini menggambarkan cara kerja penerbitan dengan video multitrack:



Panduan OBS

1. Unduh dan instal OBS Studio:

- a. Windows: Video multitrack didukung mulai dari OBS Studio 30.2.
 - b. macOS: Video multitrack didukung mulai dari OBS Studio 31.1 Beta (khusus Apple Silicon).
 - c. Unduh di: <https://obsproject.com/unduh>.
2. Klik Pengaturan. Di bagian Stream panel Pengaturan, pilih Amazon IVS dari dropdown Layanan.
 3. Untuk Server, biarkan pengaturan sebagai Otomatis.
 4. Untuk Stream Key, masukkan streamKey dari konfigurasi ingest.
 5. Di bawah bagian Video Multitrack, periksa Aktifkan Video Multitrack.
 6. Di panel Video, atur Resolusi Basis (Canvas Resolution) dan Output (Scaled) yang diinginkan. Streaming real-time IVS mendukung input hingga 720p. Jika Anda melebihi batas ini, aliran Anda akan terputus.

Ketika video multitrack diaktifkan, pengaturan seperti jumlah trek video, bitrate mereka, dan interval keyframe secara otomatis dikonfigurasi berdasarkan kemampuan perangkat.

7. Pilih Mulai Streaming.

Penerbitan dengan FFmpeg

Anda dapat menggunakan FFmpeg untuk mempublikasikan video langsung dan audio ke streaming real-time IVS melalui RTMP. FFmpeg adalah proyek sumber terbuka gratis yang terdiri dari rangkaian perpustakaan perangkat lunak dan alat yang komprehensif untuk memproses video, audio, dan konten multimedia lainnya.

Contoh perintah berikut menerbitkan aliran yang menyertakan pola warna dan nada:

```
ffmpeg \  
-re \  
-f lavfi -i testsrc=d=300:s=1280x720:r=60,format=yuv420p \  
-f lavfi -i sine=f=440:b=4:d=300 \  
-c:v libx264 \  
-b:v 2500k \  
-g 60 -bf 0 \  
-profile:v baseline \  
-preset veryfast \  
-tune zerolatency \  
-x264opts sliced-threads=0 \  
-c:a aac \  
-ac 2 \  
-b:a 160k \  

```

```
-ar 48000 \  
-f flv \  
rtmps://$INGEST_ENDPOINT/app/$STREAM_KEY
```

Dalam contoh, ganti `$INGEST_ENDPOINT` dan `$STREAM_KEY` dengan nilai Anda sendiri dari konsol IVS atau API.

Konfigurasi ini memenuhi [Spesifikasi Media yang Didukung](#) untuk streaming real-time IVS, termasuk video H.264 (profil dasar, tidak ada bingkai-B, tidak ada utas yang diiris) dan audio AAC.

Pencernaan Pribadi ke Tahapan

Anda dapat mempublikasikan aliran RTMP (S) dan E-RTMP (S) ke tahap dari sumber daya di dalam VPC Amazon Anda atau dari Direct Connect, dengan menggunakan titik akhir VPC antarmuka. Ini memungkinkan koneksi pribadi antara VPC dan IVS Anda, menjaga lalu lintas tertelan dalam jaringan AWS. Untuk mengatur dan mengonfigurasi titik akhir VPC antarmuka untuk IVS, lihat IVS [Private Ingest di Panduan Pengguna Streaming Latensi Rendah IVS](#).

Penerbitan IVS 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, yang menawarkan alternatif (untuk SDK siaran IVS) untuk publikasi desktop. Streamer yang lebih berpengalaman yang terbiasa dengan OBS mungkin lebih menyukainya karena fitur produksinya yang canggih, seperti transisi adegan, mixing 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 enkoder perangkat keras, SDK siaran IVS mungkin bukanlah sebuah pilihan. Namun, jika enkoder mendukung WHIP, Anda masih dapat memublikasikan langsung dari enkoder ke IVS.

Persyaratan WHIP:

- Penawaran SDP Anda harus menyertakan trek video H.264, bahkan jika Anda hanya menerbitkan audio. Jika penawaran tidak termasuk trek video, koneksi akan ditolak.

- Titik akhir WHIP global (<https://global.whip.live-video.net>) mengembalikan Pengalihan Sementara 307. Klien WHIP harus menangani 307 pengalihan dengan benar dan mempertahankan header dalam permintaan yang dialihkan, seperti yang dipersyaratkan oleh spesifikasi WHIP.

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 1.
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 sangat cepat dan Tune ke zerolatenency, 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.

Catatan: Kami mengetahui masalah kualitas (seperti pembekuan video intermiten) yang dapat terjadi dengan WHIP di OBS. Ini biasanya muncul ketika jaringan penyiar tidak stabil. Kami merekomendasikan pengujian WHIP di OBS sebelum menggunakannya untuk streaming langsung produksi. Menurunkan bitrate siaran Anda juga dapat membantu mengurangi terjadinya masalah ini.

Replikasi Peserta IVS | Streaming Waktu Nyata

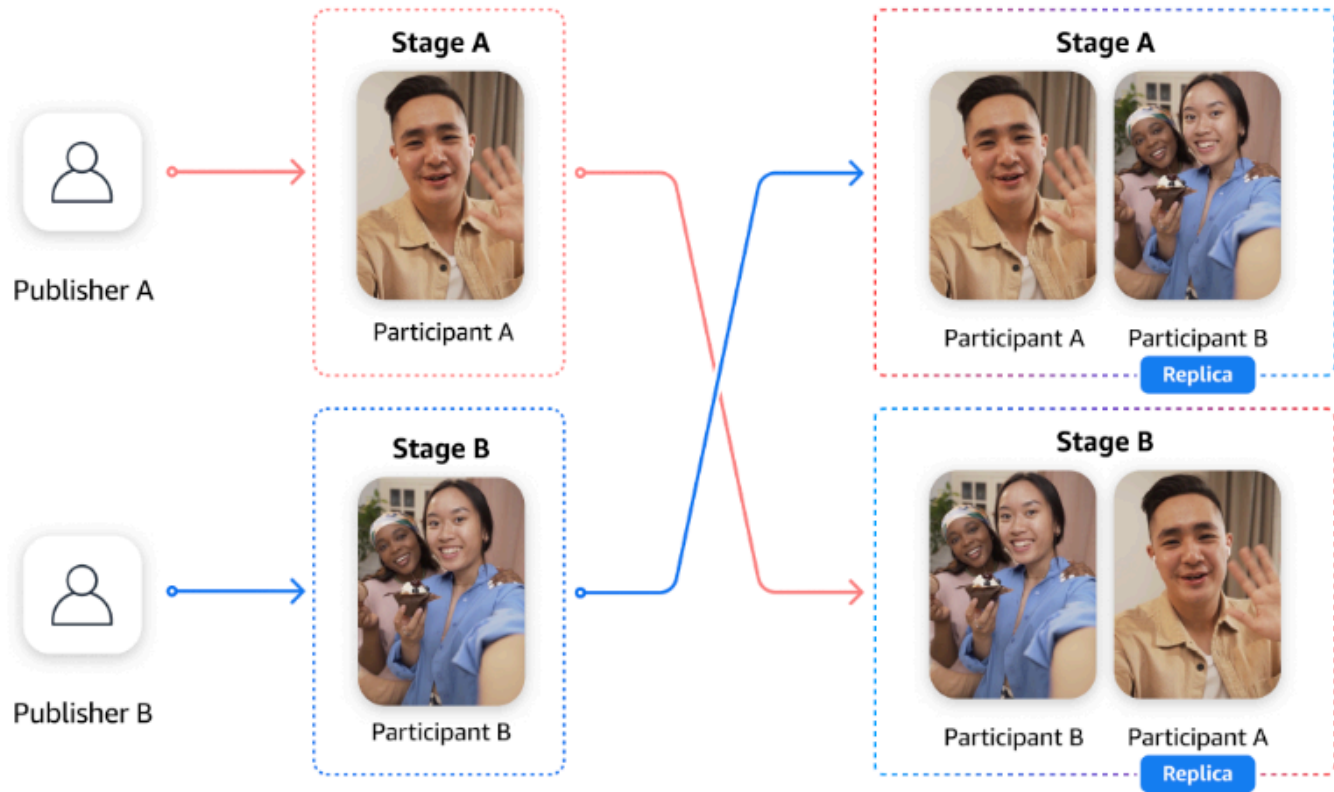
Replikasi peserta memungkinkan Anda untuk menyalin peserta dari satu tahap ke tahap lainnya. Ini berguna ketika Anda ingin peserta yang sama muncul dalam beberapa tahap pada saat yang sama, memungkinkan interaksi lintas tahap.

Kasus penggunaan umum dalam aplikasi streaming langsung sosial adalah kompetisi, sering disebut Mode VS, di mana dua streamer dicocokkan sementara sehingga mereka dapat berinteraksi satu sama lain secara real time, sementara pemirsa setiap aliran dapat melihat kedua streamer.

Konsep Kunci:

- Tahap sumber — Tahap di mana peserta awalnya bergabung, yang digunakan sebagai sumber replikasi.
- Tahap tujuan — Tahap di mana peserta direplikasi.
- Peserta yang direplikasi — Peserta dalam tahap yang direplikasi ke satu atau lebih tahap tujuan.
- Peserta replika — Peserta dalam tahap tujuan yang direplikasi dari tahap lain (tahap sumber).

Menggunakan Replikasi Peserta



Prasyarat

Untuk menggunakan replikasi peserta, Anda harus memiliki setidaknya dua tahap yang sudah dibuat. Misalnya, dalam skenario di atas, kami memiliki dua penerbit aktif:

1. Peserta A, terhubung ke tahap A
2. Peserta B, terhubung ke tahap B

Kami akan mereplikasi peserta A ke tahap B dan peserta B ke tahap A sementara, untuk mendukung head-to-head kompetisi.

Mulai Replikasi Peserta

Untuk mereplikasi peserta, gunakan `StartParticipantReplication` operasi. Anda harus memanggil ini sekali untuk setiap arah replikasi.

Replikasi peserta A ke tahap B:

```
aws ivs-realtime start-participant-replication \  
  --source-stage-arn arn:aws:ivs:us-east-1:123456789012:stage/StageA \  
  --destination-stage-arn arn:aws:ivs:us-east-1:123456789012:stage/StageB \  
  --participant-id participant-a-id \  
  --reconnect-window-seconds 10
```

Replikasi peserta B ke tahap A:

```
aws ivs-realtime start-participant-replication \  
  --source-stage-arn arn:aws:ivs:us-east-1:123456789012:stage/StageB \  
  --destination-stage-arn arn:aws:ivs:us-east-1:123456789012:stage/StageA \  
  --participant-id participant-b-id \  
  --reconnect-window-seconds 10
```

Setelah replikasi dimulai, peserta tetap direplikasi sampai Anda secara eksplisit menghentikannya menggunakan operasi `StopParticipantReplication`. Peserta yang direplikasi yang terputus dan kemudian terhubung kembali dalam interval yang ditentukan oleh `reconnectWindowSeconds` akan secara otomatis muncul lagi di tahap sumber dan tujuan. Nilai default untuk `reconnectWindowSeconds` adalah 0.

Hentikan Replikasi Peserta

Untuk menghentikan replikasi, hubungi `StopParticipantReplication` operasi.

Hentikan replikasi peserta A dari tahap A ke tahap B:

```
aws ivs-realtime stop-participant-replication \  
  --source-stage-arn arn:aws:ivs:us-east-1:123456789012:stage/StageA \  
  --destination-stage-arn arn:aws:ivs:us-east-1:123456789012:stage/StageB \  
  --participant-id participant-a-id
```

Hentikan replikasi peserta B dari tahap B ke tahap A:

```
aws ivs-realtime stop-participant-replication \  
  --source-stage-arn arn:aws:ivs:us-east-1:123456789012:stage/StageB \  
  --destination-stage-arn arn:aws:ivs:us-east-1:123456789012:stage/StageA \  
  --participant-id participant-b-id
```

Service Quotas IVS | 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 berdasarkan per akun di wilayah AWS tertentu.

Peningkatan Kuota Layanan

Untuk kuota yang dapat disesuaikan, Anda dapat meminta kenaikan tarif melalui [AWS konsol](#). Gunakan konsol juga untuk melihat informasi tentang kuota layanan.

Kuota tingkat panggilan API tidak dapat disesuaikan.

Kuota Tingkat Panggilan API

Tipe operasi	Operasi	Default
Komposisi	GetComposition	5 TPS
Komposisi	ListCompositions	5 TPS
Komposisi	StartComposition	5 TPS
Komposisi	StopComposition	5 TPS
IngestConfiguration	CreateIngestConfiguration	5 TPS
IngestConfiguration	DeleteIngestConfiguration	5 TPS
IngestConfiguration	GetIngestConfiguration	5 TPS
IngestConfiguration	ListIngestConfigurations	5 TPS

Tipe operasi	Operasi	Default
IngestConfiguration	UpdateIngestConfiguration	5 TPS
MediaEncoder	CreateEncoderConfiguration	5 TPS
MediaEncoder	DeleteEncoderConfiguration	5 TPS
MediaEncoder	GetEncoderConfiguration	5 TPS
MediaEncoder	ListEncoderConfigurations	5 TPS
ParticipantReplication	ListParticipantReplicas	5 TPS
ParticipantReplication	StartParticipantReplication	5 TPS
ParticipantReplication	StopParticipantReplication	5 TPS
PublicKey	DeletePublicKey	3 TPS
PublicKey	GetPublicKey	3 TPS
PublicKey	ImportPublicKey	3 TPS
PublicKey	ListPublicKeys	3 TPS
Stage	CreateParticipantToken	50 TPS
Stage	CreateStage	5 TPS
Stage	DeleteStage	5 TPS
Stage	DisconnectParticipant	5 TPS
Stage	GetParticipant	5 TPS
Stage	GetStage	5 TPS
Stage	GetStageSession	5 TPS
Stage	ListStages	5 TPS

Tipe operasi	Operasi	Default
Stage	UpdateStage	5 TPS
Stage	ListParticipants	5 TPS
Stage	ListParticipantEvents	5 TPS
Stage	ListStageSessions	5 TPS
StorageConfiguration	CreateStorageConfiguration	5 TPS
StorageConfiguration	DeleteStorageConfiguration	5 TPS
StorageConfiguration	GetStorageConfiguration	5 TPS
StorageConfiguration	ListStorageConfigurations	5 TPS
Tanda	ListTagsForResource	10 TPS
Tag	TagResource	10 TPS
Tag	UntagResource	10 TPS

Kuota Lainnya

Sumber Daya atau Fitur	Default	Dapat disesuaikan	Deskripsi
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.

Sumber Daya atau Fitur	Default	Dapat disesuaikan	Deskripsi
Komposisi	20	Ya	Sumber daya Komposisi bersamaan maksimum per akun.
Komposisi per tahap	5	Ya	Sumber daya Komposisi bersamaan maksimum per tahap.
Replikasi peserta bersamaan	5	Tidak	Jumlah maksimum replikasi bersamaan per peserta di semua tahapan di Wilayah AWS.
Penerbit bersamaan	1.000	Ya	Jumlah maksimum peserta yang dapat mempublikasikan di semua tahapan di Wilayah AWS.
Langganan bersamaan	20.000	Ya	Jumlah maksimum publisher-to-subscriber koneksi simultan di semua tahapan di Wilayah AWS.
EncoderConfigurations	20	Ya	Jumlah maksimum sumber EncoderConfiguration daya per akun.
IngestConfigurations	100	Ya	Jumlah maksimum sumber IngestConfiguration daya per akun.
Bitrate unduhan peserta	8,5 Mbps	Tidak	Bitrate unduhan agregat maksimum di semua langganan peserta.

Sumber Daya atau Fitur	Default	Dapat disesuaikan	Deskripsi
Peserta mempublikasikan bitrate	8,5 Mbps	Tidak	Bit maksimum per detik yang dapat dialirkan ke panggung.
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.
PublicKeys	3	Tidak	Jumlah maksimum kunci publik, per AWS Wilayah.
Peserta panggung (penerbit)	12	Tidak	Jumlah maksimum peserta yang dapat mempublikasikan ke panggung sekaligus.
Peserta panggung (pelanggan)	10.000	Ya	Jumlah maksimum peserta yang dapat berlangganan panggung sekaligus.
Tahapan	1.000	Ya	Jumlah maksimum tahapan, per AWS Wilayah.
StorageConfigurations	5	Ya	Jumlah maksimum sumber StorageConfiguration daya per akun.

Optimasi Streaming Waktu Nyata IVS

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.

IVS SDKs memungkinkan Anda untuk mengkonfigurasi bitrate maksimum, framerate, dan resolusi aliran. Ketika kemacetan jaringan terjadi untuk penerbit, SDK secara otomatis menyesuaikan dan menurunkan kualitas video dengan menurunkan bitrate, framerate, dan resolusi. Di Android dan iOS, dimungkinkan untuk memilih preferensi degradasi saat kemacetan ditemui. Perilaku yang sama berlaku apakah Anda mengaktifkan pengkodean berlapis dengan simulcast atau mempertahankan konfigurasi default.

Streaming Adaptif: Pengkodean Berlapis dengan Simulcast

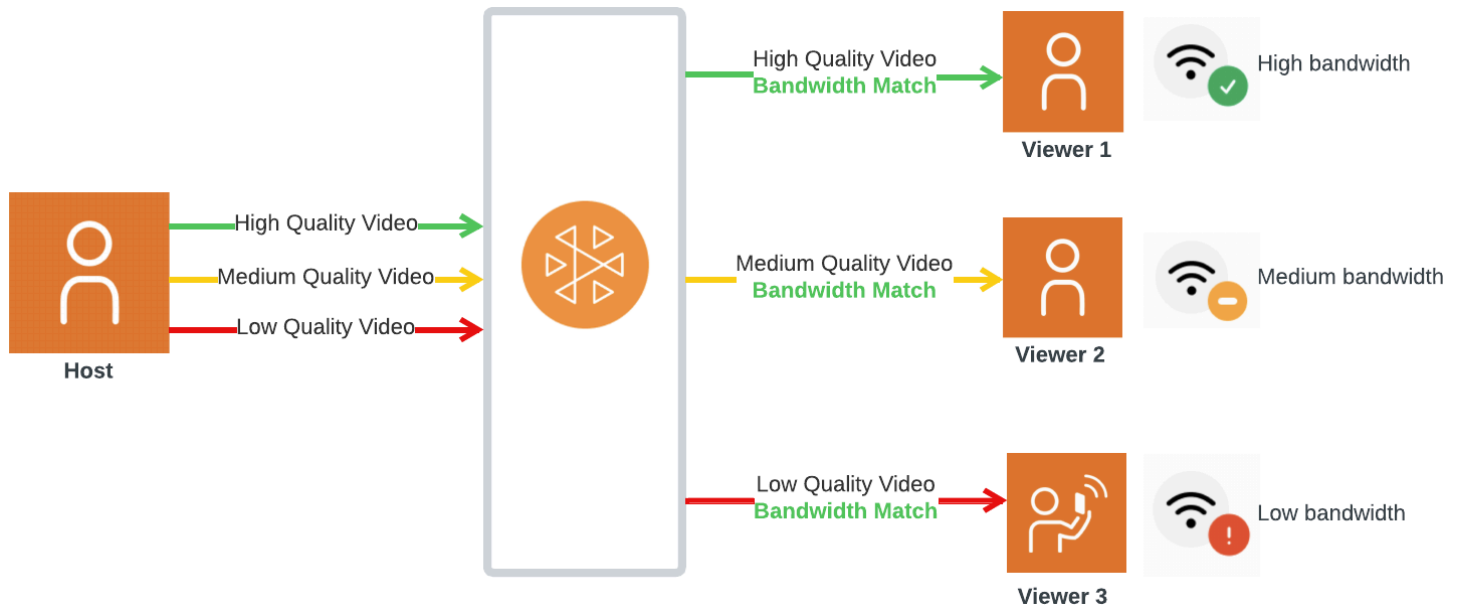
Fitur ini hanya didukung dalam versi klien berikut:

- iOS dan Android 1.18.0+
- Web 1.12.0+

Saat menggunakan [siaran real-time](#) IVS SDKs, penerbit dapat 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 dan Edge (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 Viewer 1 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)
Lapisan tinggi (atau kustom): <ul style="list-style-type: none"> • Bitrate maks: 900.000 bps • Framerate: 15 fps 	Lapisan tinggi (atau kustom): <ul style="list-style-type: none"> • Bitrate maks: 1.700.000 bps • Framerate: 30 fps
Lapisan tengah: tidak ada (tidak diperlukan, karena perbedaan antara bitrate lapisan tinggi dan rendah di ponsel sempit)	Lapisan tengah: <ul style="list-style-type: none"> • Bitrate maks: 700.000 bps • Framerate: 20 fps
Lapisan rendah:	Lapisan rendah:

Seluler (Android, iOS)	Web (Chrome)
<ul style="list-style-type: none"> • Bitrate maks: 100.000 bps • Framerate: 15 fps 	<ul style="list-style-type: none"> • Bitrate maks: 200.000 bps • Framerate: 15 fps

Resolusi Lapisan

Resolusi lapisan menengah dan rendah secara otomatis diperkecil dari lapisan tinggi, untuk mempertahankan rasio aspek yang sama.

Lapisan menengah dan rendah dikecualikan jika resolusinya terlalu dekat dengan lapisan di atas. Misalnya, jika resolusi yang dikonfigurasi adalah 320x180, SDK juga tidak akan mengirim lapisan beresolusi lebih rendah.

Tabel di bawah ini menunjukkan resolusi lapisan yang dihasilkan untuk resolusi yang dikonfigurasi berbeda. Nilai yang tercantum dalam orientasi lanskap tetapi dapat diterapkan secara terbalik untuk konten potret.

Resolusi Masukan	Resolusi Lapisan Keluaran: Seluler	Resolusi Lapisan Keluaran: Web
720p (1280x720)	Hai (1280x720)	Hai (1280x720)
	Rendah (320x180)	Pertengahan (640x360)
		Rendah (320x180)
540p (960x540)	Hai (960x540)	Hai (960x540)
	Rendah (320x180)	Rendah (320x180)
360p (640x360)	Hai (640x360)	Hai (640x360)
	Rendah (360x180)	Rendah (360x180)
270p (480x270)	Hai (480x270)	Hai (480x270)
180p (320x180)	Hai (320x180)	Hai (320x180)

Untuk resolusi input khusus yang tidak dipetakan di atas, Anda dapat menghitungnya [menggunakan alat berikut](#).

Mengkonfigurasi Layered Encoding dengan Simulcast (Publisher)

Untuk menggunakan encoding berlapis dengan simulcast, Anda [harus mengaktifkan fitur pada klien](#). Jika Anda mengaktifkannya, Anda akan melihat peningkatan penggunaan bandwidth unggahan oleh penerbit, berpotensi dengan pembekuan video yang lebih sedikit untuk pemirsa.

Android

```
// Enable Simulcast
StageVideoConfiguration config = new StageVideoConfiguration();
config.simulcast.setEnabled(true);

ImageLocalStageStream cameraStream = new ImageLocalStageStream(frontCamera, config);

// Other Stage implementation code
```

iOS

```
// Enable Simulcast
let config = IVSLocalStageStreamVideoConfiguration()
config.simulcast.enabled = true

let cameraStream = IVSLocalStageStream(device: camera, configuration: config)

// Other Stage implementation code
```

Web

```
// Enable Simulcast
let cameraStream = new LocalStageStream(cameraDevice, {
  simulcast: { enabled: true }
})

// Other Stage implementation code
```

[Untuk informasi terperinci tentang mengonfigurasi setiap lapisan, lihat “Mengonfigurasi Pengkodean Berlapis \(Penerbit\)” di setiap panduan SDK siaran: Android, iOS, dan Web.](#)

Mengonfigurasi Layered Encoding dengan Simulcast (Subscriber)

Untuk mengonfigurasi lapisan apa yang diterima oleh pelanggan, lihat bagian “Layered Encoding with Simulcast” di panduan SDK streaming real-time:

- [SDK Siaran Android](#)
- [SDK Siaran iOS](#)
- [SDK Siaran Web](#)

Dengan konfigurasi pelanggan, dimungkinkan untuk menentukan `InitialLayerPreference` ini menentukan kualitas video apa yang dikirimkan pada awalnya, serta `preferredLayerForStream`, yang pada gilirannya menentukan lapisan apa yang dipilih selama pemutaran video. Ada peristiwa dan metode aliran untuk memberi tahu ketika lapisan berubah, perubahan adaptasi, atau pemilihan lapisan dibuat.

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

```
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
```

Mengubah Buffer Jitter Pelanggan MinDelay

Untuk mengubah penundaan minimum buffer jitter bagi peserta yang sedang berlangganan, kustom `subscribeConfiguration` dapat digunakan. Buffer jitter menentukan berapa banyak paket yang disimpan sebelum pemutaran dimulai. Penundaan minimum mewakili target untuk jumlah minimum data yang harus disimpan. Mengubah penundaan minimum dapat membantu pemutaran menjadi lebih tangguh saat menghadapi masalah paket loss/connection .

Pengorbanan saat meningkatkan ukuran buffer jitter adalah bahwa hal itu juga akan meningkatkan penundaan sebelum pemutaran dimulai. Meningkatkan penundaan minimum memberikan lebih banyak ketahanan, dengan mengorbankan waktu yang memengaruhi video. Perhatikan bahwa meningkatkan penundaan minimum selama pemutaran memiliki efek yang serupa: pemutaran akan berhenti sebentar untuk memungkinkan buffer jitter terisi.

Jika diperlukan lebih banyak ketahanan, sebaiknya mulai dengan preset penundaan minimum `MEDIUM` dan pengaturan konfigurasi berlangganan sebelum pemutaran dimulai.

Perhatikan bahwa penundaan minimum hanya diterapkan jika peserta hanya berlangganan. Jika peserta menerbitkan sendiri, penundaan minimum tidak diterapkan. Hal ini dilakukan untuk memastikan bahwa beberapa penerbit dapat berbicara satu sama lain tanpa penundaan tambahan.

Contoh di bawah ini menggunakan preset penundaan minimum. MEDIUM Lihat dokumentasi referensi SDK untuk semua nilai yang mungkin.

Web

```
const strategy = {
  subscribeConfiguration: (participant) => {
    return {
      jitterBuffer: {
        minDelay: JitterBufferMinDelay.MEDIUM
      }
    }

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

Android

```
@Override
public SubscribeConfiguration subscribeConfigurationForParticipant(@NonNull Stage stage,
    @NonNull ParticipantInfo participantInfo) {
    SubscribeConfiguration config = new SubscribeConfiguration();

    config.jitterBuffer.setMinDelay(JitterBufferConfiguration.JitterBufferDelay.MEDIUM());

    return config;
}
```

iOS

```
func stage(_ stage: IVSStage, subscribeConfigurationForParticipant participant:
    IVSParticipantInfo) -> IVSSubscribeConfiguration {
    let config = IVSSubscribeConfiguration()

    try! config.jitterBuffer.setMinDelay(.medium())

    return config
}
```

}

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 .

Persyaratan Jaringan | Streaming Waktu Nyata

Amazon IVS real-time streaming bergantung pada WebRTC dan WebSocket protokol untuk transmisi media dan data. Untuk memastikan pengalaman yang mulus, tujuan dan pelabuhan yang tercantum di bawah ini harus dapat diakses. Pembatasan apa pun pada lalu lintas masuk atau keluar ke tujuan ini dapat mengganggu fungsionalitas streaming real-time IVS.

Anda dapat menggunakan [alat uji jaringan](#) ini untuk memastikan bahwa jaringan Anda dikonfigurasi dengan benar dan bahwa lalu lintas ke dan dari tujuan dan port yang diperlukan tidak diblokir.

Biasa

Destinasi	Port
*.live-video.net	TCP:443

Media

Secara default, streaming real-time IVS bergantung pada UDP untuk transmisi media guna memastikan latensi rendah dan streaming berkinerja tinggi. Jika UDP diblokir untuk peserta berlangganan media, TCP digunakan sebagai fallback. Fallback TCP tidak didukung untuk penerbitan, jadi jika lalu lintas UDP benar-benar diblokir, penerbitan akan gagal.

Destinasi	Port
Semua subnet yang terdaftar di bawah IVS_REALTIME layanan di ip-ranges.json harus dapat diakses, terlepas dari Wilayah AWS mereka atau region yang Anda pilih. Peserta dapat terhubung ke subnet apa pun secara otomatis. Lihat Solusi Global , Kontrol Regional untuk detailnya.	UDP:3478
	UDP:443
	TCP: 3478 (Mundur)
	TCP:443

Biaya IVS | Streaming Waktu Nyata

Lihat [halaman Harga IVS](#) untuk detail tentang biaya IVS.

- Berlangganan dan menerbitkan ke tahapan — Berlangganan dan menerbitkan menghabiskan sumber daya, dan Anda akan dikenakan tarif per jam untuk waktu Anda terhubung ke panggung.
- Rekaman — Rekaman peserta individu tidak dikenakan biaya Amazon IVS tambahan, sementara rekaman komposit dikenakan biaya untuk tarif per jam untuk video yang dikodekan. Kedua opsi perekaman menimbulkan penyimpanan S3 standar dan biaya permintaan. Thumbnail tidak dikenakan biaya IVS tambahan.
- Replikasi peserta — Peserta replika ditagih sama dengan peserta reguler.

Misalnya, Anda memiliki dua tahap, Tahap A dengan Peserta A dan Tahap B dengan Peserta B. Anda dikenakan biaya untuk dua peserta.

Jika Peserta A direplikasi ke Tahap B, Anda sekarang memiliki tiga peserta yang terhubung (Peserta A, Peserta B, dan replika Peserta A). Selama replikasi, Anda dikenakan biaya untuk tiga peserta.

Informasi lebih lanjut ada di halaman Harga IVS.

Sumber Daya dan Dukungan IVS | Streaming Waktu Nyata

Dokumen ini mencantumkan sumber daya untuk membantu mendukung penggunaan streaming real-time Amazon IVS.

Demo dan Sumber Daya Lainnya

<https://ivs.rocks/> adalah situs khusus untuk menelusuri konten yang dipublikasikan (demo, contoh kode, posting blog), memperkirakan biaya, dan pengalaman IVS melalui demo langsung. Untuk contoh demo dan kode, lihat <https://ivs.rocks/examples>.

[Halaman Amazon IVS di situs Komunitas DEV](#) memiliki banyak demo dan posting blog. Misalnya, [Memulai dengan Amazon Interactive Video Service](#) adalah serangkaian artikel tentang penggunaan IVS, untuk pemula. Artikel memberikan step-by-step panduan IVS APIs dengan demo interaktif yang tertanam dalam posting. Semua demo dapat dijalankan langsung di posting itu sendiri melalui tertanam CodePen.

Situs [AWS Blogs](#) memiliki banyak posting blog IVS tentang berbagai topik. Filter untuk IVS dengan memilih Produk atau solusi > Layanan Media > Layanan Video Interaktif Amazon di sisi kanan halaman.

Pada GitHub, demo streaming real-time IVS untuk iOS dan Android menunjukkan kepada pengembang cara menggunakan IVS untuk membangun aplikasi konten real-time yang menarik: social-user-generated



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 tempat tamu atau versus (VS). Petunjuk tentang cara menerapkan backend yang diperlukan dan membangun aplikasi tersedia di repositori berikut: GitHub

- 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: [-serverless-demo/ https://github.com/aws-samples/ amazon-ivs-real-time](https://github.com/aws-samples/amazon-ivs-real-time-serverless-demo/)

Support

[AWS Support Center](#) menawarkan berbagai paket yang menyediakan akses ke alat dan keahlian untuk mendukung AWS solusi Anda. 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 Anda, pilih paket dukungan yang paling sesuai dengan kasus AWS penggunaan Anda.

[AWS Premium Support](#) adalah one-on-one saluran dukungan respons cepat untuk membantu Anda membangun dan menjalankan aplikasi. AWS

[AWS Re:post](#) adalah situs Tanya Jawab berbasis komunitas bagi pengembang untuk mendiskusikan pertanyaan teknis yang terkait dengan Amazon IVS.

[Kontak AWS](#) memiliki tautan untuk pertanyaan nonteknis tentang penagihan atau akun Anda. Untuk pertanyaan teknis, gunakan forum diskusi atau tautan dukungan di atas.

Glosarium IVS

Lihat juga [Glosarium AWS](#). Pada tabel di bawah ini, LL adalah singkatan dari streaming latensi rendah IVS; RT, streaming waktu nyata IVS.

Istilah	Deskripsi	LL	RT	Obrolan
AAC	Pengodean Audio Tingkat Lanjut. AAC adalah standar pengodean audio untuk kompresi audio digital lossy. Dirancang untuk menjadi penerus MP3 format, AAC umumnya mencapai kualitas suara yang lebih tinggi daripada MP3 pada bitrate yang sama. AAC telah distandardisasi oleh ISO dan IEC sebagai bagian dari spesifikasi MPEG-2 dan MPEG-4.	✓	✓	
Streaming laju bit adaptif	Streaming Laju Bit Adaptif (ABR) memungkinkan pemutar IVS untuk beralih ke laju bit yang lebih rendah ketika kualitas koneksi menurun, dan untuk beralih kembali ke laju bit yang lebih tinggi ketika kualitas koneksi meningkat.	✓		
Streaming adaptif	Lihat Enkode berlapis dengan simulcast .		✓	
Pengguna administratif	Seorang pengguna AWS dengan akses administratif ke sumber daya dan layanan yang tersedia di akun AWS. Lihat Terminologi di Panduan Pengguna Pengaturan AWS.	✓	✓	✓
ARN	Amazon Resource Name , pengidentifikasi unik untuk sumber daya AWS. Format ARN spesifik bergantung pada tipe sumber daya. Untuk format ARN yang digunakan oleh sumber daya IVS, lihat di Referensi Otorisasi Layanan.	✓	✓	✓

Istilah	Deskripsi	LL	RT	Obrolan
Rasio aspek	Mendeskripsikan rasio lebar bingkai dengan tinggi bingkai. Misalnya, 16:9 adalah rasio aspek yang sesuai dengan resolusi Full HD atau 1080p.	✓	✓	
Mode audio	Suatu konfigurasi audio preset atau kustom yang dioptimalkan untuk berbagai tipe pengguna perangkat seluler dan perangkat yang mereka gunakan. Lihat SDK Siaran IVS: Mode Audio Seluler (Streaming Waktu Nyata) .		✓	
AVC, H.264, MPEG-4 Part 10	Pengodean Audio Tingkat Lanjut, disebut juga dengan H.264 atau MPEG-4 Part 10, sebuah standar kompresi video untuk kompresi video digital lossy.	✓	✓	
Penggantian latar belakang	Tipe filter kamera yang memungkinkan kreator streaming langsung untuk mengubah latar belakang mereka. Lihat Penggantian Latar Belakang di SDK Siaran IVS: Filter Kamera Pihak Ketiga (Streaming Waktu Nyata).		✓	
Laju bit	Suatu metrik streaming untuk jumlah bit yang ditransmisikan atau diterima per detik.	✓	✓	
Siaran, penyiar	Istilah lain untuk streaming , streamer .	✓		
Buffering	Suatu kondisi yang terjadi ketika perangkat pemutaran tidak dapat mengunduh konten sebelum konten seharusnya diputar. Buffering dapat terjadi dalam beberapa cara: konten dapat berhenti dan memulai secara acak (disebut juga dengan tersendat), konten dapat berhenti dalam jangka waktu yang lama (juga dikenal sebagai freezing), 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 daftar putar rentang byte, 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. Daftar putar tersebut dibuat sebagai tambahan untuk daftar putar HLS. Lihat Daftar Putar Rentang Byte di Rekam Otomatis ke Amazon S3 (Streaming Latensi Rendah).</p>	✓		
CBR	<p>Laju Bit Konstan, sebuah metode kontrol laju untuk enkoder yang mempertahankan laju bit agar konsisten di seluruh pemutaran video, terlepas dari kejadian selama siaran. Jeda dalam tindakan mungkin ditambahkan untuk mencapai laju bit yang diinginkan, dan puncak mungkin dikuantisasi dengan menyesuaikan kualitas encode agar cocok dengan laju bit target. Kami sangat menyarankan penggunaan CBR, alih-alih VBR.</p>	✓	✓	
CDN	<p>Jaringan Pengiriman Konten atau Jaringan Distribusi Konten, sebuah solusi yang didistribusikan secara geografis yang mengoptimalkan pengiriman konten seperti streaming video dengan membawanya lebih dekat dengan tempat pengguna berada.</p>	✓		

Istilah	Deskripsi	LL	RT	Obrolan
Channel	Suatu sumber daya IVS yang menyimpan konfigurasi untuk streaming, termasuk ingest server , kunci streaming , URL pemutaran , dan opsi perekaman. Streamer menggunakan kunci streaming yang terkait dengan saluran untuk memulai siaran. Semua metrik dan peristiwa yang dihasilkan selama siaran dikaitkan dengan sumber daya saluran.	✓		
Tipe saluran	Menentukan resolusi dan laju bingkai yang diizinkan untuk saluran . Lihat Tipe Saluran di Referensi API Streaming Latensi Rendah IVS.	✓		
Pembuatan log obrolan	Sebuah opsi lanjutan yang dapat diaktifkan dengan mengaitkan konfigurasi pembuatan log dengan ruang obrolan .			✓
Ruang obrolan	Sumber daya IVS yang menyimpan konfigurasi untuk sesi obrolan, termasuk fitur opsional seperti Handler Tinjauan Pesan dan Pembuatan Log Obrolan . Lihat Langkah 2: Buat Ruang Obrolan di Memulai Obrolan IVS.			✓
Komposisi di sisi klien	Menggunakan perangkat host untuk mencampur aliran audio dan video dari peserta stage, kemudian mengirimkannya sebagai streaming komposit ke saluran IVS. Hal ini memungkinkan kontrol yang lebih leluasa atas tampilan komposisi dengan mengorbankan pemanfaatan sumber daya klien yang lebih tinggi dan risiko yang lebih tinggi atas masalah stage atau host yang berdampak pada pemirsa. Lihat juga komposisi di sisi server .	✓	✓	
CloudFront	Layanan CDN yang disediakan oleh Amazon.	✓		

Istilah	Deskripsi	LL	RT	Obrolan
CloudTrail	Sebuah layanan AWS untuk mengumpulkan, memantau, menganalisis, dan mempertahankan peristiwa serta aktivitas akun dari AWS dan sumber eksternal. Lihat Mencatat Panggilan API IVS dengan AWS CloudTrail .	✓	✓	✓
CloudWatch	Sebuah layanan AWS untuk memantau aplikasi, dengan merespons perubahan performa, mengoptimalkan penggunaan sumber daya, dan memberikan wawasan tentang kondisi operasional. Anda dapat menggunakannya CloudWatch untuk memantau metrik IVS; lihat Memantau Streaming Waktu Nyata IVS dan Pemantauan Streaming Latensi Rendah IVS .	✓	✓	✓
Komposisi	Proses menggabungkan streaming audio dan video dari lebih dari satu sumber menjadi satu streaming.	✓	✓	
Pipeline komposisi	Suatu urutan langkah-langkah pemrosesan yang diperlukan untuk menggabungkan banyak streaming dan mengkode streaming yang dihasilkan.	✓	✓	
Kompresi	Proses encode informasi dengan menggunakan bit yang lebih kecil dari representasi asli. Kompresi tertentu dapat bersifat 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 tidak terlalu penting.	✓	✓	

Istilah	Deskripsi	LL	RT	Obrolan
Bidang kontrol	Menyimpan informasi tentang sumber daya IVS seperti saluran , stage , atau ruang obrolan dan menyediakan antarmuka untuk membuat serta mengelola sumber daya ini. Bidang kontrol bersifat regional (berdasarkan wilayah AWS).	✓	✓	✓
CORS	Cross-Origin Resource Sharing, sebuah fitur AWS yang mengizinkan aplikasi web klien yang dimuat di dalam satu domain untuk berinteraksi dengan sumber daya seperti bucket S3 dalam domain yang berbeda. Akses dapat dikonfigurasi berdasarkan header, metode HTTP, dan domain asal. Lihat Penggunaan cross-origin resource sharing (CORS) - Amazon Simple Storage Service di Panduan Pengguna Amazon Simple Storage Service.	✓		
Sumber audio khusus	Antarmuka yang disediakan oleh IVS Broadcast SDK yang memungkinkan aplikasi menyediakan input audio sendiri, alih-alih terbatas pada mikrofon bawaan perangkat.		✓	
Sumber gambar kustom	Suatu antarmuka yang disediakan oleh SDK Siaran IVS yang memungkinkan sebuah aplikasi untuk memberikan input gambarnya sendiri, alih-alih terbatas pada kamera preset.	✓	✓	
Pemesanan peserta khusus	Memungkinkan pemosisian peserta tahap di tata letak kisi dan PiP berdasarkan nilai atribut khusus dalam token peserta.		✓	
Bidang data	Infrastruktur yang membawa data dari ingest ke egress. Infrastruktur ini beroperasi berdasarkan konfigurasi yang dikelola di bidang kontrol dan tidak terbatas pada wilayah AWS.	✓	✓	✓

Istilah	Deskripsi	LL	RT	Obrolan
Enkoder, encode	Proses konversi konten video dan audio ke dalam format digital, cocok untuk streaming. Encode dapat berbasis perangkat keras atau perangkat lunak.	✓	✓	
E-RTMP	Protokol RTMP yang disempurnakan . IVS mendukung fitur E-RTMP yang diperlukan untuk video multitrack .	✓		
Peristiwa	Pemberitahuan otomatis yang diterbitkan oleh IVS ke layanan AmazonEventBridge pemantauan. Suatu peristiwa merepresentasikan perubahan status atau kondisi dari sumber daya streaming seperti stage atau pipeline komposisi . Lihat Menggunakan Amazon EventBridge dengan Streaming Latensi Rendah IVS dan Menggunakan Amazon EventBridge dengan Streaming Waktu Nyata IVS .	✓	✓	✓
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 kemudian terhubung kembali dalam interval yang ditentukan dalam konfigurasi perekaman saluran . Banyaknya streaming yang dihasilkan dianggap sebagai satu siaran dan digabungkan menjadi satu streaming rekaman tunggal. Lihat Menggabungkan Streaming Terfragmentasi di Rekam Otomatis ke Amazon S3 (Streaming Latensi Rendah).	✓		

Istilah	Deskripsi	LL	RT	Obrolan
Laju bingkai	Suatu metrik streaming untuk jumlah bingkai video yang ditransmisikan atau diterima per detik.	✓	✓	
HLS	Streaming Langsung HTTP (HLS), sebuah protokol komunikasi streaming laju bit adaptif berbasis HTTP yang digunakan untuk mengirimkan streaming IVS kepada pemirsa.	✓		
Daftar putar HLS	Daftar segmen media yang menyusun streaming. Daftar putar HLS standar terdiri dari file media 10 detik. HLS juga mendukung daftar putar rentang byte yang lebih terperinci.	✓		
Host	Pengguna real-time yang membuat panggung.		✓	
IAM	Identity and Access Management, sebuah layanan AWS yang memungkinkan pengguna untuk mengelola identitas dan akses ke layanan serta sumber daya AWS dengan aman, termasuk IVS.	✓	✓	✓
Ingest	Proses IVS untuk menerima streaming video dari host atau penyiar untuk diproses atau dikirimkan ke pemirsa atau peserta lain.	✓	✓	
Ingest server	Menerima streaming video dan mengirimkannya ke sistem transkode, saat transmux atau transkode streaming dilakukan ke HLS untuk dikirim ke pemirsa. Ingest server adalah komponen IVS khusus yang menerima streaming untuk saluran , bersama dengan protokol penyerapan (RTMP , RTMPS). Lihat informasi tentang pembuatan saluran di Memulai Streaming Latensi Rendah IVS .	✓		

Istilah	Deskripsi	LL	RT	Obrolan
Video interlaced	Mentransmisikan dan menampilkan hanya baris ganjil atau genap dari bingkai berikutnya untuk menciptakan kesan penggandaan laju bingkai tanpa menghabiskan bandwidth ekstra. Kami tidak menyarankan menggunakan video interlaced terkait 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.	✓	✓	✓
Keyframe, bingkai delta, interval keyframe	Keyframe (disebut juga dengan intra-coded atau i-frame) adalah bingkai penuh dari gambar dalam sebuah video. Bingkai berikutnya, bingkai delta (disebut juga dengan predicted atau p-frame), hanya berisi informasi yang telah berubah. Keyframe akan muncul berkali-kali dalam streaming , bergantung pada interval keyframe yang ditentukan dalam enkoder.	✓	✓	
Lambda	Sebuah 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 invokasi, atau berdasarkan jadwal. Misalnya, Obrolan IVS memanfaatkan fungsi Lambda guna mengaktifkan tinjauan pesan untuk ruang obrolan .	✓	✓	✓

Istilah	Deskripsi	LL	RT	Obrolan
Latensi, glass-to-glass latensi	<p>Keterlambatan dalam transfer data. IVS menentukan rentang latensi sebagai berikut:</p> <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>	✓	✓	
Enkode berlapis dengan simulcast	Memungkinkan enkode dan publikasi simultan untuk banyak streaming video dengan tingkat kualitas yang berbeda. Lihat Streaming Adaptif: Enkode Berlapis dengan Simulcast di Optimisasi Streaming Waktu Nyata.		✓	
Handler tinjauan pesan	Memungkinkan pelanggan IVS Chat untuk pesan obrolan review/filter pengguna secara otomatis sebelum dikirim ke ruang obrolan . Handler ini diaktifkan dengan mengaitkan fungsi Lambda dengan ruang obrolan. Lihat Membuat Fungsi Lambda di Handler Tinjauan Pesan Obrolan.			✓
Mixer	Fitur IVS Mobile Broadcast SDKs yang mengambil beberapa sumber audio dan video dan menghasilkan satu output. Fitur ini mendukung manajemen elemen video dan audio di layar yang merepresentasikan sumber-sumber seperti kamera, mikrofon, tangkapan layar, dan audio serta video yang dihasilkan oleh aplikasi. Output kemudian dapat dialirkan ke IVS. Lihat Mengonfigurasi Sesi Siaran untuk Mixing di SDK Siaran IVS: Panduan Mixing (Streaming Latensi Rendah).	✓		

Istilah	Deskripsi	LL	RT	Obrolan
Streaming banyak host	<p>Menggabungkan streaming dari banyak host ke dalam satu streaming. Hal ini dapat dicapai dengan menggunakan komposisi di sisi klien atau sisi server.</p> <p>Dengan streaming banyak host, banyak skenario yang dapat dilakukan, seperti mengundang pemirsa ke stage untuk Tanya Jawab, kompetisi antar-host, obrolan video, dan percakapan antara host di depan banyak pemirsa.</p>		✓	
Video Multitrack	<p>Memungkinkan alat perangkat lunak penyiar untuk menyandikan dan mengalirkan beberapa kualitas video langsung dari komputer bertenaga GPU. Lihat Video Multitrack Amazon IVS.</p>	✓		
Daftar putar multivarian	<p>Indeks dari semua streaming 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>	✓		
OBS	<p>Open Broadcaster Software, perangkat lunak gratis dan sumber terbuka untuk perekaman video dan streaming langsung. OBS menawarkan alternatif (untuk SDK siaran IVS) untuk publikasi desktop. Streamer yang lebih berpengalaman yang terbiasa dengan OBS mungkin lebih menyukainya karena fitur produksinya yang canggih, seperti transisi adegan, mixing audio, dan grafik overlay.</p>	✓	✓	
Peserta	<p>Pengguna real-time yang terhubung ke panggung sebagai penerbit atau pelanggan.</p>		✓	

Istilah	Deskripsi	LL	RT	Obrolan
Peserta memesan	Urutan di mana peserta tahap diposisikan dalam tata letak grid dan PiP.		✓	
Token peserta	Mengautentikasi peserta acara waktu nyata saat mereka bergabung dengan suatu stage . Token peserta juga mengontrol apakah peserta dapat mengirim video ke stage.		✓	
Token pemutaran , pasangan kunci pemutaran	<p>Mekanisme otorisasi yang memungkinkan pelanggan membatasi pemutaran video di saluran privat. Token pemutaran dihasilkan dari pasangan kunci pemutaran.</p> <p>Pasangan kunci pemutaran adalah pasangan kunci publik-privat yang digunakan untuk menandatangani dan memvalidasi token otorisasi pemirsa untuk pemutaran. Lihat Membuat atau Mengimpor Kunci Pemutaran IVS dalam Menyiapkan Saluran Pribadi IVS dan lihat operasi Playback Key Pair di Referensi API Latensi Rendah IVS.</p>	✓		
URL pemutaran	Mengidentifikasi alamat yang digunakan oleh pemirsa guna memulai pemutaran untuk 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 pembuatan saluran di Memulai Streaming Latensi Rendah IVS .	✓		
Saluran privat	Memungkinkan pelanggan membatasi akses ke streaming mereka dengan menggunakan mekanisme otorisasi berdasarkan token pemutaran . Lihat Alur Kerja untuk Saluran Pribadi IVS dalam Menyiapkan Saluran Pribadi IVS .	✓		

Istilah	Deskripsi	LL	RT	Obrolan
Ingest pribadi	Mengaktifkan koneksi pribadi yang aman antara VPC Amazon dan IVS Anda menggunakan titik akhir VPC antarmuka yang didukung oleh AWS. PrivateLink Lihat IVS Private Ingest .	✓		
Video progresif	Mentransmisikan dan menampilkan semua baris dari setiap bingkai secara berurutan. Kami menyarankan penggunaan video progresif selama semua stage siaran.	✓	✓	
Penerbit	Peserta acara real-time yang mempublikasikan and/or audio video ke panggung. Lihat Apa itu Streaming Waktu Nyata IVS .		✓	
Kuota	Kuota ini merupakan jumlah maksimum sumber daya atau operasi layanan IVS untuk akun AWS Anda. Batas ini adalah per akun AWS, kecuali disebutkan lain dalam tabel. Semua kuota diberlakukan per wilayah. Lihat Titik akhir dan kuota Amazon Interactive Video Service di Panduan Referensi Umum AWS.	✓	✓	✓

Istilah	Deskripsi	LL	RT	Obrolan
Wilayah	<p>Menyediakan akses ke layanan AWS yang secara fisik ada di wilayah geografis tertentu. Wilayah memberikan toleransi kesalahan, stabilitas, serta ketahanan, dan juga dapat mengurangi latensi. Dengan Wilayah, Anda dapat membuat sumber daya redundan yang tetap tersedia dan tidak terpengaruh oleh pemadaman wilayah.</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 mana pun, kecuali jika 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>	✓	✓	✓
Resolusi	Menjelaskan jumlah piksel dalam satu bingkai video, misalnya, Full HD atau 1080p mendefinisikan sebuah bingkai dengan 1920 x 1080 piksel.	✓	✓	
Pengguna root	Pemilik akun AWS. Pengguna root memiliki akses ke semua layanan dan sumber daya AWS dalam akun AWS.	✓	✓	✓
RTMP, RTMPS	Protokol Perpesanan Waktu Nyata, sebuah standar industri untuk mentransmisikan audio, video, dan data melalui jaringan. RTMPS adalah RTMP versi aman, yang berjalan melalui koneksi Keamanan Lapisan Pengangkutan (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 banyak objek dalam bucket S3.	✓		

Istilah	Deskripsi	LL	RT	Obrolan
SDK	<p>Kit Pengembangan Perangkat Lunak, suatu kumpulan pustaka bagi developer yang membangun aplikasi dengan IVS.</p> <p>SDK IVS Player adalah untuk pemutaran aliran IVS. Ini memanfaatkan arsitektur IVS dan dioptimalkan untuk pemutaran latensi rendah IVS. Ada pemutar IVS SDKs untuk web, Android, dan iOS.</p> <p>IVS Broadcast SDK adalah untuk pengembang yang sedang membangun aplikasi dengan IVS. SDK ini memanfaatkan arsitektur IVS dan terus ditingkatkan, bersama dengan IVS. Sebagai SDK siaran asli, ini dirancang untuk meminimalkan dampak kinerja pada aplikasi Anda dan pada perangkat yang digunakan pengguna mengakses aplikasi. Ada siaran IVS SDKs untuk web, Android, dan iOS, untuk latensi rendah dan streaming real-time.</p>	✓	✓	✓
Segmentasi selfie	<p>Memungkinkan penggantian latar belakang dalam streaming langsung dengan menggunakan solusi khusus klien yang menerima gambar kamera sebagai input dan mengembalikan mask yang memberikan skor kepercayaan untuk setiap piksel gambar, yang menunjukkan apakah gambar itu di latar depan atau latar belakang. Lihat Penggantian Latar Belakang di SDK Siaran IVS: Filter Kamera Pihak Ketiga (Streaming Waktu Nyata).</p>		✓	

Istilah	Deskripsi	LL	RT	Obrolan
Penentuan versi semantik	Sebuah format versi dalam bentuk Major.Minor.Patch. Perbaiki bug yang tidak memengaruhi peningkatan API versi tambalan, API yang kompatibel ke belakang additions/changes meningkatkan versi minor, dan perubahan API yang tidak kompatibel ke belakang meningkatkan versi utama.	✓	✓	✓
Komposisi di sisi server	Menggunakan server IVS untuk mixing audio dan video dari peserta stage kemudian mengirimkan mixing video ini ke saluran IVS untuk menjangkau pemirsa yang lebih besar atau menyimpannya dalam bucket S3 . Komposisi di sisi server mengurangi beban klien sehingga meningkatkan ketahanan siaran dan memungkinkan penggunaan bandwidth yang lebih efisien. Lihat juga komposisi di sisi klien .		✓	
Kuota layanan	Sebuah layanan AWS yang membantu mengelola kuota Anda untuk banyak layanan AWS dari satu lokasi. Selain mencari nilai kuota, Anda juga dapat meminta peningkatan kuota dari konsol Service Quotas.	✓	✓	✓
Peran tertaut layanan	Tipe peran IAM unik yang ditautkan secara langsung ke layanan AWS. Peran tertaut layanan dibuat secara otomatis oleh IVS dan mencakup semua izin yang diperlukan layanan untuk memanggil layanan AWS lainnya atas nama Anda, misalnya, untuk mengakses bucket S3 . Lihat Menggunakan Peran Tertaut Layanan untuk IVS di Keamanan IVS.	✓		

Istilah	Deskripsi	LL	RT	Obrolan
Stage	Sumber daya IVS yang merepresentasikan ruang virtual tempat peserta acara waktu nyata dapat bertukar video dalam waktu nyata. Lihat Membuat Panggung dengan Rekaman Peserta Opsional dalam Memulai Streaming Waktu Nyata IVS.		✓	
Sesi stage	Dimulai ketika peserta pertama bergabung dengan stage dan berakhir beberapa menit setelah peserta terakhir berhenti memublikasikan ke stage. Suatu stage berdurasi panjang mungkin memiliki banyak sesi selama masa tayangnya.		✓	
Streaming	Data yang merepresentasikan konten video atau audio yang dikirim secara terus-menerus dari sumber ke tujuan.	✓	✓	
Kunci streaming	Pengidentifikasi yang ditetapkan oleh IVS saat Anda membuat saluran ; pengidentifikasi ini digunakan untuk mengotorisasi streaming ke saluran. Perlakukan kunci streaming seperti suatu rahasia, karena siapa pun yang memilikinya dapat melakukan streaming ke saluran. Lihat Memulai Streaming Latensi Rendah IVS .	✓		

Istilah	Deskripsi	LL	RT	Obrolan
Kekurangan streaming	<p>Suatu penundaan atau penghentian pengiriman streaming ke IVS. Hal ini terjadi ketika IVS tidak menerima jumlah bit yang diharapkan yang diiklankan oleh perangkat encode yang akan dikirim selama jangka waktu tertentu. Terjadinya kekurangan streaming menyebabkan peristiwa kekurangan streaming.</p> <p>Dari sudut pandang pemirsa, kekurangan streaming dapat tampak seperti video terputus-putus, mengalami buffering, atau freezing. Kekurangan streaming dapat berlangsung singkat (kurang dari 5 detik) atau lama (beberapa menit), bergantung pada situasi spesifik yang mengakibatkan kekurangan streaming. Lihat Apa itu Kekurangan Streaming di FAQ Pemecahan Masalah.</p>	✓	✓	
Streamer	Seseorang atau suatu perangkat yang mengirimkan streaming video atau audio ke IVS.	✓	✓	
Pelanggan	Peserta acara real-time yang menerima and/or audio video dari penerbit panggung. Lihat Apa itu Streaming Waktu Nyata IVS .		✓	
Tag	Sebuah label metadata yang Anda tetapkan ke sumber daya AWS. Tanda dapat membantu Anda mengidentifikasi dan mengelola sumber daya AWS. Pada halaman arahan dokumentasi IVS , lihat “Penandaan” di dokumentasi API IVS mana pun (untuk streaming waktu nyata, streaming latensi rendah, atau obrolan).	✓	✓	✓

Istilah	Deskripsi	LL	RT	Obrolan
Filter kamera pihak ketiga	Komponen perangkat lunak yang dapat diintegrasikan dengan SDK Siaran IVS agar aplikasi dapat memproses gambar sebelum memberikannya ke SDK Siaran sebagai sumber gambar kustom . Sebuah filter kamera pihak ketiga dapat memproses gambar dari kamera, menerapkan efek filter, dll.	✓	✓	
Thumbnail	Sebuah gambar berukuran kecil yang diambil dari streaming. Secara default, thumbnail dihasilkan setiap 60 detik, tetapi interval yang lebih pendek juga dapat dikonfigurasi. Resolusi thumbnail bergantung pada tipe saluran . Lihat Merekam Konten di Rekam Otomatis ke Amazon S3 (Streaming Latensi Rendah).	✓		
Metadata berwaktu	<p>Metadata yang terikat pada stempel waktu tertentu dalam suatu aliran. Metadata ini dapat ditambahkan secara terprogram menggunakan API IVS dan menjadi terkait dengan bingkai tertentu. Hal ini memastikan bahwa semua pemirsa menerima metadata pada titik yang relatif sama terhadap streaming.</p> <p>Metadata berwaktu dapat digunakan untuk memicu tindakan pada klien seperti memperbarui statistik tim selama acara olahraga. Lihat Menyematkan Metadata dalam Streaming Video.</p>	✓		

Istilah	Deskripsi	LL	RT	Obrolan
Pertukaran Token	Antarmuka yang disediakan oleh IVS Broadcast SDK yang memungkinkan peningkatan atau penurunan kemampuan token peserta dan memperbarui atribut token, tanpa mengharuskan peserta untuk terhubung kembali. Ini memungkinkan skenario seperti co-hosting, di mana peserta dapat memulai dengan kemampuan berlangganan saja dan kemudian membutuhkan kemampuan publikasi.		✓	
Transkode	Mengonversi video dan audio dari satu format ke format lainnya. Streaming masuk dapat ditranskode ke berbagai format pada berbagai laju bit dan resolusi, untuk mendukung berbagai perangkat pemutaran dan kondisi jaringan.	✓	✓	
Transmuxing	Pengemasan ulang sederhana dari streaming ingest ke IVS, tanpa encode ulang streaming video. “Transmux” adalah kependekan dari transcode multiplexing, sebuah proses yang mengubah format file audio dan/atau video sambil menyimpan beberapa atau semua streaming asli. Transmuxing mengonversi ke berbagai format kontainer tanpa mengubah isi file. Berbeda dari transkode .	✓	✓	

Istilah	Deskripsi	LL	RT	Obrolan
Streaming varian	<p>Suatu set encode siaran yang sama dalam berbagai tingkat kualitas. Setiap streaming varian diekode sebagai daftar putar HLS terpisah. Indeks streaming varian yang tersedia disebut sebagai daftar putar multivarian.</p> <p>Setelah pemutar IVS menerima daftar putar multivarian dari IVS, pemutar IVS kemudian dapat memilih antara streaming varian selama pemutaran, berganti-ganti dengan mulus saat kondisi jaringan berubah.</p>	✓		
VBR	<p>Variable Bitrate, metode kontrol laju untuk enkoder yang menggunakan laju bit dinamis yang berubah sepanjang pemutaran, bergantung pada tingkat detail yang diperlukan. Kami sangat tidak menyarankan penggunaan VBR karena masalah kualitas video; gunakan CBR sebagai gantinya.</p>	✓	✓	
Tayang	<p>Sesi tampilan unik yang secara aktif mengunduh atau memutar file media. Tayangan adalah dasar untuk kuota tayangan bersamaan.</p> <p>Suatu tayangan dimulai ketika sesi penayangan memulai pemutaran video. Suatu tayangan diakhiri ketika sesi penayangan mengakhiri pemutaran video. Pemutaran adalah satu-satunya indikator pemirsa; heuristik interaksi seperti tingkat audio, fokus tab peramban, dan kualitas video bukan tidak dipertimbangkan. Saat menghitung tayangan, IVS tidak mempertimbangkan legitimasi pemirsa individu atau mencoba menghapus duplikat pemirsa lokal, seperti banyak pemutar video pada satu mesin. Lihat Kuota Lainnya di Service Quotas (Streaming Latensi Rendah).</p>	✓		

Istilah	Deskripsi	LL	RT	Obrolan
Pemirsa	Seseorang yang menerima streaming dari IVS.	✓		
WebRTC	<p>Komunikasi Waktu Nyata Web, sebuah proyek sumber terbuka yang menyediakan peramban web dan aplikasi seluler dengan komunikasi waktu nyata. 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 JavaScript APIs secara reguler di semua browser utama atau sebagai pustaka untuk klien asli, seperti Android dan iOS.</p>	✓	✓	

Istilah	Deskripsi	LL	RT	Obrolan
WHIP	<p>WebRTC-HTTP Ingestion Protocol, protokol berbasis HTTP yang memungkinkan konsumsi konten berbasis WebRTC ke dalam layanan streaming. and/or CDNs WHIP adalah draf IETF yang dikembangkan untuk menstandardisasi penyerapan WebRTC.</p> <p>WHIP memungkinkan kompatibilitas dengan perangkat lunak seperti OBS, yang menawarkan alternatif (untuk SDK siaran IVS) untuk publikasi desktop. Streamer yang lebih berpengalaman yang terbiasa dengan OBS mungkin lebih menyukainya karena fitur produksinya yang canggih, seperti transisi adegan, mixing audio, dan grafik overlay</p> <p>WHIP juga bermanfaat dalam situasi ketika penggunaan SDK siaran IVS tidak dimungkinkan atau tidak disukai. Misalnya, dalam pengaturan yang melibatkan enkoder perangkat keras, SDK siaran IVS mungkin bukanlah sebuah pilihan. Namun, jika enkoder mendukung WHIP, Anda masih dapat memublikasikan langsung dari enkoder ke IVS.</p> <p>Lihat Dukungan IVS WHIP (Streaming Waktu Nyata).</p>		✓	
WSS	<p>WebSocket Secure, protokol untuk membangun WebSockets melalui koneksi TLS terenkripsi. Protokol 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 IVS | Streaming Waktu Nyata

Tabel berikut menjelaskan perubahan penting pada dokumentasi untuk Amazon IVS Real-Time Streaming. Kami sering memperbarui dokumentasi, untuk rilis baru dan untuk mengatasi umpan balik yang Anda kirimkan kepada kami.

Perubahan Panduan Pengguna Streaming Waktu Nyata

Perubahan	Deskripsi	Tanggal
Siaran SDK: Web 1.33.0	Nomor versi dan tautan artefak yang diperbarui dalam panduan SDK low-latency-streaming siaran: Web. Lihat juga Catatan Rilis .	Maret 12, 2026
Siaran SDK: Android 1.40.0, iOS 1.40.0	Nomor versi dan tautan artefak yang diperbarui dalam panduan SDK real-time-streaming siaran: Android dan iOS. Lihat juga Catatan Rilis .	Maret 12, 2026
EventBridge	Untuk empat peristiwa (Kegagalan Tujuan, Kegagalan Sesi, Kegagalan Mulai Perekaman, Kegagalan Akhir Perekaman), kami memperbarui deskripsi dan menambahkan <code>error_code</code> bidang dan tabel kode kesalahan dan alasan.	Februari 27, 2026
Siaran SDK: Android 1.39.0, iOS 1.39.0	Nomor versi dan tautan artefak yang diperbarui dalam panduan SDK real-time-streaming siaran: Android dan iOS. Lihat juga Catatan Rilis .	Februari 13, 2026

Untuk integrasi iOS, CocoaPods tidak lagi digunakan. Perubahan dokumentasi terkait dibuat di:

- [Memulai Streaming Waktu Nyata IVS](#) - “Langkah 4: Integrasikan SDK Siaran IVS” > “iOS”
- [Panduan SDK Siaran iOS](#) — “Instal Perpustakaan”

[Siaran SDK: Web 1.32.0](#)

[Nomor versi dan tautan artefak yang diperbarui dalam panduan SDK low-latency-streaming siaran: Web.](#) Lihat juga [Catatan Rilis](#).

Februari 12, 2026

[Service Quotas](#)

Menambahkan nilai TPS untuk tiga ParticipantReplication operasi (List/Start/Stop).

Januari 30, 2026

[Siaran SDK: Android 1.38.0, iOS 1.38.0](#)

[Nomor versi dan tautan artefak yang diperbarui dalam panduan SDK real-time-streaming siaran: Android dan iOS.](#) Lihat juga [Catatan Rilis](#).

Januari 13, 2026

[Siaran SDK: Android 1.37.1](#)

[Nomor versi dan tautan artefak yang diperbarui dalam panduan SDK real-time-streaming siaran: Android.](#) Lihat juga [Catatan Rilis](#).

Desember 11, 2025

[Broadcast SDK - Sumber Audio Kustom](#)

Ditambahkan halaman baru ini.

Desember 10, 2025

Pertukaran token peserta	<p>Kami menambahkan halaman Token Exchange baru di bawah IVS Broadcast SDK.</p> <p>Di Menggunakan Amazon EventBridge dengan IVS, kami menambahkan acara Pembaruan Tahap IVS, Token Exchanged. Kami juga menambahkan contoh Token Exchanged di Contoh: Pembaruan Tahap.</p> <p>Perubahan API dijelaskan dalam tabel Referensi API.</p>	Desember 9, 2025
Siaran SDK: Web 1.31.0	<p>Nomor versi dan tautan artefak yang diperbarui dalam panduan SDK low-latency-streaming siaran: Web. Lihat juga Catatan Rilis.</p>	Desember 5, 2025
Siaran SDK: Android 1.37.0, iOS 1.37.0	<p>Nomor versi dan tautan artefak yang diperbarui dalam panduan SDK real-time-streaming siaran: Android dan iOS. Lihat juga Catatan Rilis.</p>	Desember 5, 2025
CloudWatch metrik	<p>Dalam Memantau Streaming Real-Time > CloudWatch Metrik, kami menambahkan banyak metrik baru DroppedFrames PublishBitrate, dan SubscribeBitrate metrik. Kami juga memperbarui beberapa deskripsi metrik.</p>	November 18, 2025

Sinkronisasi IPR	Dalam Rekaman Peserta Individu, kami menambahkan Sinkronisasi Rekaman Beberapa Peserta .	November 7, 2025
Komposisi Sisi Server	Menambahkan Masalah dan Solusi yang Diketahui .	Oktober 30, 2025
Siaran SDK: Web 1.30.0	Nomor versi dan tautan artefak yang diperbarui dalam panduan SDK low-latency-streaming siaran: Web . Lihat juga Catatan Rilis .	Oktober 30, 2025
Siaran SDK: Android 1.36.0, iOS 1.36.0	Nomor versi dan tautan artefak yang diperbarui dalam panduan SDK real-time-streaming siaran: Android dan iOS . Lihat juga Catatan Rilis . Kami juga menambahkan bagian baru pada “Embed Messages”: Android dan iOS .	Oktober 30, 2025
Komposisi Sisi Server	Siklus Hidup Komposisi yang Diperbarui (saat komposisi melakukan shutdown otomatis).	Oktober 27, 2025
RTMP	Ditambahkan Penerbitan dengan FFmpeg .	Oktober 27, 2025
Perbarui kuota untuk Komposisi	Di Service Quotas > Kuota Lainnya , kami memperbarui kuota untuk “sumber daya Komposisi bersamaan maksimum per akun” dari 5 menjadi 20.	Oktober 14, 2025

Pembaruan SDK Siaran Web	Menulis ulang bagian tentang Penerbitan & Berlangganan dengan SDK Siaran Web IVS > Dapatkan Statistik WebRTC.	Oktober 3, 2025
CloudWatch Metrik-metrik	Untuk ConcurrentPublishers dan ConcurrentSubscriptions metrik, memperbarui deskripsi dan menghapus dimensi.	Oktober 2, 2025
Siaran SDK: Web 1.29.0	Nomor versi dan tautan artefak yang diperbarui dalam panduan SDK low-latency-streaming siaran: Web. Lihat juga Catatan Rilis .	Oktober 2, 2025
Siaran SDK: Android 1.35.0, iOS 1.35.0	Nomor versi dan tautan artefak yang diperbarui dalam panduan SDK real-time-streaming siaran: Android dan iOS. Lihat juga Catatan Rilis .	Oktober 2, 2025
CloudWatch Metrik-metrik	Menambahkan beberapa DownloadPacketLoss metrik lagi.	September 23, 2025
Pemesanan peserta khusus SSC	Membuat berbagai perubahan pada Komposisi Sisi Server , termasuk menambahkan participantOrderAttribute dan “Pemesanan Peserta Kustom.” Perubahan API dijelaskan dalam tabel Referensi API .	September 16, 2025

Siaran SDK: Android 1.34.0, iOS 1.34.0	Nomor versi dan tautan artefak yang diperbarui dalam panduan SDK real-time-streaming siaran: Android dan iOS. Lihat juga Catatan Rilis .	September 11, 2025
Cerapan Pribadi ke Tahapan	Bagian baru untuk rilis antarmuka VPC endpoint.	September 10, 2025
Siaran SDK: Web 1.28.0	Nomor versi dan tautan artefak yang diperbarui dalam panduan SDK low-latency-streaming siaran: Web. Lihat juga Catatan Rilis .	September 4, 2025
Siaran SDK: iOS	Menambahkan perbaikan SDK siaran iOS 1.32.1 ke Catatan Rilis terbaru untuk SDK siaran iOS (1.33.0).	Agustus 21, 2025
Siaran SDK: Web	Pembaruan di Memulai> Impor , baik Menggunakan Tag Skrip dan Menggunakan npm.	Agustus 8, 2025
Siaran SDK: Web 1.27.0	Nomor versi dan tautan artefak yang diperbarui dalam panduan SDK low-latency-streaming siaran: Web. Lihat juga Catatan Rilis .	Agustus 7, 2025

[Siaran SDK: Android 1.33.0, iOS 1.33.0](#)

[Nomor versi dan tautan artefak yang diperbarui dalam panduan SDK real-time-streaming siaran: Android dan iOS.](#) Lihat juga [Catatan Rilis](#).

Agustus 7, 2025

Dalam Panduan Siaran iOS, kami menambahkan “Direkomendasikan: Integrasi kan SDK Pemain (Swift Package Manager)” dan memperbarui informasi yang ada tentang CocoaPods integrasi.

[Broadcast SDK: Perangkat Campuran](#)

Ini adalah dokumen baru. ([Perangkat Campuran](#) identik dalam Panduan Pengguna Streaming Latensi Rendah IVS dan Panduan Pengguna Streaming Waktu Nyata IVS.)

Juli 28, 2025

[Siaran SDK: Android 1.32.2](#)

[Nomor versi dan tautan artefak yang diperbarui dalam panduan SDK real-time-streaming siaran: Android.](#) Lihat juga [Catatan Rilis](#).

Juli 25, 2025

[Tambahkan batas untuk replikasi peserta bersamaan](#)

Di Service Quotas > [Other Quotas](#), kami menambahkan kuota untuk “replikasi peserta bersamaan.”

Juli 15, 2025

[Siaran SDK: Android 1.32.1, iOS 1.32.1](#)

[Nomor versi dan tautan artefak yang diperbarui dalam panduan SDK real-time-streaming siaran: Android dan iOS.](#) Lihat juga [Catatan Rilis](#).

Juli 10, 2025

[Siaran SDK: Web 1.26.0](#)

[Nomor versi dan tautan artefak yang diperbarui dalam panduan SDK low-latency-streaming siaran: Web.](#) Lihat juga [Catatan Rilis](#).

Juli 7, 2025

Juga:

- Dalam [Menangani Masalah Jaringan](#), kami memperbarui contoh.
- Di [StageErrors](#), kami menambahkan informasi tentang kesalahan GAGAL dan kami menambahkan kesalahan STAGE_DISCONNECTED dan PARTICIPANT_DISCONNECTED.

[Tambahkan batas untuk penerbit dan langganan bersamaan](#)

Di Service Quotas > [Other Quotas](#), kami menambahkan kuota untuk “penerbit bersamaan” dan “langganan bersamaan.”

Juni 23, 2025

Dalam Memantau Streaming Real-Time > [CloudWatch Metrik](#), kami menambahkan metrik untuk ConcurrentPublishers dan ConcurrentSubscriptions

[Dalam Broadcast Web SDK Guide](#), kami memperbarui entri [tabel STAGE_AT_CAPACITY](#) di [Error Handling > Stage Errors](#).

[Dukungan menelan video multitrack E-RTMP](#)

Di [Stream Ingest > Protokol yang Didukung](#), kami menambahkan dukungan untuk video multitrack E-RTMP (Enhanced RTMP).

Juni 20, 2025

Di [IVS RTMP Publishing](#), kami menambahkan informasi tentang streaming video multitrack E-RTMP menggunakan siaran dan OBS Studio. real-time-streaming SDKs

Dalam [Memantau Streaming Real-Time > CloudWatch Metrik](#), kami menambahkan PublishFramerate metrik dengan Tahap, Peserta SimulcastLayer, dan MediaType dimensi. Kami juga memperbarui nilai SimulcastLayer dimensi (“no-rid” dan “disabled” diganti dengan “none”).

[Enkripsi SSE-S3](#)

Kami menambahkan informasi baru ke:

Juni 19, 2025

- Rekaman Peserta Individu — Dalam [1. Buat Bucket S3](#)
- [Komposisi Sisi Server - Dalam Memulai dengan Komposisi Sisi Server IVS > Prasyarat](#)

Siaran SDK: Web 1.25.1	Nomor versi dan tautan artefak yang diperbarui dalam panduan SDK low-latency-streaming siaran: Web. Lihat juga Catatan Rilis .	Juni 16, 2025
Siaran SDK: Web 1.25.0	Nomor versi dan tautan artefak yang diperbarui dalam panduan SDK low-latency-streaming siaran: Web. Lihat juga Catatan Rilis .	12 Juni 2025
Siaran SDK: Android 1.31.0, iOS 1.31.0	Nomor versi dan tautan artefak yang diperbarui dalam panduan SDK real-time-streaming siaran: Android dan iOS. Lihat juga Catatan Rilis .	12 Juni 2025
B-frame dalam aliran menelan	Di Stream Ingest > Spesifikasi Media yang Didukung , kami memperbarui informasi tentang B-frame.	30 Mei 2025

[Replikasi peserta](#)

Rilis awal dari fungsi baru ini.
Lihat perubahan dokumentasi ini:

29 Mei 2025

- Memulai Amazon IVS — Pada [Langkah 2: Buat Panggung dengan Perekaman Peserta Opsional, instruksi konsol dan tangkapan layar yang diperbarui dan ditambahkan recordParticipantR eplicas](#) ke respons CLI untuk membuat panggung dengan perekaman peserta individu.
- Memantau Streaming Waktu Nyata — Dalam [CloudWatch Metrik: Streaming Waktu Nyata IVS](#), menambahkan catatan tentang replikasi peserta.
- EventBridge — Dalam [Contoh: Pembaruan Tahap](#), menambahkan dua acara, Mulai Replikasi Peserta dan Akhir Replikasi Peserta. Juga diperbarui Peserta Diterbitkan, Peserta Tidak Dipublikasikan, dan Kesalahan Publikasi Peserta.
- [Replikasi Peserta](#) — Dokumen baru ini mencakup ikhtisar dan instruksi CLI.

- [Biaya](#) — Dokumen baru ini mencakup biaya yang terkait dengan streaming real-time IVS.

Perubahan API dijelaskan dalam tabel [Referensi API](#).

Siaran SDK: Android 1.30.1	Nomor versi dan tautan artefak yang diperbarui dalam panduan SDK real-time-streaming siaran: Android. Lihat juga Catatan Rilis .	26 Mei 2025
Siaran SDK: Web 1.24.0	Nomor versi dan tautan artefak yang diperbarui dalam panduan SDK low-latency-streaming siaran: Web. Lihat juga Catatan Rilis .	15 Mei 2025
Siaran SDK: Android 1.30.0, iOS 1.30.0	Nomor versi dan tautan artefak yang diperbarui dalam panduan SDK real-time-streaming siaran: Android dan iOS. Lihat juga Catatan Rilis .	15 Mei 2025
Siaran SDK: Web 1.23.1	Nomor versi dan tautan artefak yang diperbarui dalam panduan SDK low-latency-streaming siaran: Web. Lihat juga Catatan Rilis .	2 Mei 2025

[Siaran SDK: Web 1.23.0](#)

[Nomor versi dan tautan artefak yang diperbarui dalam panduan SDK low-latency-streaming siaran: Web](#). Lihat juga [Catatan Rilis](#).

17 April 2025

Kami juga menambahkan informasi dan contoh tambahan ke “Configuring Layered Encoding (Publisher)” di [Web Broadcast SDK Guide](#).

[Siaran SDK: Android 1.29.0, iOS 1.29.0](#)

[Nomor versi dan tautan artefak yang diperbarui dalam panduan SDK real-time-streaming siaran: Android dan iOS](#). Lihat juga [Catatan Rilis](#).

17 April 2025

Kami juga menambahkan informasi dan contoh tambahan untuk “Mengkonfigurasi Layered Encoding (Publisher)” di Panduan SDK Siaran Android [dan iOS](#).

[Service Quotas](#)

Menambahkan kuota untuk “Komposisi per tahap.”

April 2, 2025

[Pengoptimalan Streaming](#)

Dalam pendahuluan, menambahkan paragraf tentang mengonfigurasi bitrate maksimum, framerate, resolusi, dan preferensi degradasi (pada seluler).

Maret 21, 2025

[Siaran SDK: Web 1.22.0](#)

[Nomor versi dan tautan artefak yang diperbarui dalam panduan SDK low-latency-streaming siaran: Web](#). Lihat juga [Catatan Rilis](#).

Maret 20, 2025

Juga, dalam pendahuluan, kami menambahkan contoh Kode Sampel lainnya (Pemutaran Sederhana). Dalam “Informasi Peningkatan Tambahan (SEI) > Memasukkan Payloads SEI,” kami menambahkan catatan tentang penggunaan memori. Dan di “Masalah & Solusi yang Diketahui,” kami menambahkan item tentang `stage.leave()`

[Siaran SDK: Android 1.28.1, iOS 1.28.1](#)

[Nomor versi dan tautan artefak yang diperbarui dalam panduan SDK real-time-streaming siaran: Android dan iOS](#). Lihat juga [Catatan Rilis](#).

Maret 20, 2025

[Siaran SDK: Android 1.27.2, iOS 1.27.2](#)

[Nomor versi dan tautan artefak yang diperbarui dalam panduan SDK real-time-streaming siaran: Android dan iOS](#). Lihat juga [Catatan Rilis](#).

Maret 19, 2025

Durasi Segmen Target	Memperbarui beberapa tangkapan layar di “Memulai dengan Streaming Waktu Nyata IVS > Langkah 2: Buat Panggung dengan Perekaman Peserta Opsional ” untuk menampilkan bidang “Durasi segmen target” yang baru.	Maret 13, 2025
Rekaman peserta individu	Ditambahkan Mengonversi Rekaman ke MP4 .	7 Maret 2025
Jahitan perekaman peserta individu	Rilis awal dari fungsi baru ini. Lihat perubahan dokumentasi ini: <ul style="list-style-type: none">• Memulai Amazon IVS — Instruksi konsol dan CLI yang diperbarui di Langkah 2: Buat Panggung dengan Perekaman Peserta Opsional.• Rekaman Peserta Individu — Menambahkan Rekaman Peserta Individu Gabungan yang Terfragmentasi.• EventBridge — Dalam Contoh: Perubahan Status Rekaman Peserta Individu, menambahkan informasi tentang konstruksi awalan S3 saat penggabungan diaktifkan untuk Perekaman Peserta Individu.	6 Maret 2025

Persyaratan WHIP	Dalam teks pengantar, ditambahkan persyaratan untuk klien WHIP untuk menangani 307 pengalihan.	Maret 5, 2025
Siaran SDK: iOS 1.27.1	Nomor versi dan tautan artefak yang diperbarui dalam panduan SDK real-time-streaming siaran: iOS. Lihat juga Catatan Rilis .	Maret 3, 2025
Siaran SDK: Web 1.21.0	Nomor versi dan tautan artefak yang diperbarui dalam panduan SDK low-latency-streaming siaran: Web. Lihat juga Catatan Rilis .	Februari 20, 2025
Siaran SDK: Android 1.27.0, iOS 1.27.0	Nomor versi dan tautan artefak yang diperbarui dalam panduan SDK real-time-streaming siaran: Android dan iOS. Lihat juga Catatan Rilis .	Februari 20, 2025
Siaran SDK: Android 1.26.0, iOS 1.26.0	Nomor versi dan tautan artefak yang diperbarui dalam panduan SDK real-time-streaming siaran: Android dan iOS. Lihat juga Catatan Rilis .	Januari 30, 2025
SDK Siaran: Web 1.20.0	Nomor versi dan tautan artefak yang diperbarui dalam panduan SDK low-latency-streaming siaran: Web. Lihat juga Catatan Rilis .	Januari 23, 2025
	Kami juga memperbarui “Informasi Tambahan Tambahan.”	

Penerbitan RTMP	Dalam Publikasikan Menggunakan Encoder RTMP , kami mencatat bahwa aliran harus menyertakan trek audio dan video atau mereka akan terputus.	Januari 21, 2025
Persyaratan Jaringan	Ditambahkan halaman tingkat atas baru ini.	Januari 21, 2025
Pengoptimalan Streaming	Berganti nama menjadi “Configuring Layered Encoding with Simulcast” menjadi “Configuring Layered Encoding with Simulcast (Publisher)” dan menambahkan “Configuring Layered Encoding with Simulcast (Subscriber).”	Desember 12, 2024
Siaran SDK: Web 1.19.0	Nomor versi dan tautan artefak yang diperbarui dalam panduan SDK low-latency-streaming siaran: Web . Lihat juga Catatan Rilis . Kami juga menambahkan “Layered Encoding with Simulcast” dan menambahkan tiga item simulcast di “Events.”	Desember 12, 2024

[Siaran SDK: Android 1.25.0, iOS 1.25.0](#)

[Nomor versi dan tautan artefak yang diperbarui dalam panduan SDK real-time-streaming siaran: Android dan iOS](#). Lihat juga [Catatan Rilis](#).

Desember 12, 2024

Di setiap panduan, kami juga:

- Ditambahkan “Dapatkan Informasi Peningkatan Tambahan.”
- Ditambahkan “Layered Encoding dengan Simulcast.”
- Menambahkan tiga item simulcast di “Renderer.”
- Dihapus “Aktifkan/Nonaktifkan Layered Encoding dengan Simulcast.”

[Konfigurasi thumbnail waktu nyata](#)

Dalam [Perekaman Peserta Individu](#) dan [Perekaman Komposit](#), contoh terbaru dan informasi metadata JSON, dan menambahkan informasi harga. Dalam [Rekaman Peserta Individu](#), tambahkan [“Rekaman Khusus Thumbnail](#).”

Desember 10, 2024

[Siaran SDK: Android 1.24.0, iOS 1.24.0](#)

[Nomor versi dan tautan artefak yang diperbarui dalam panduan SDK real-time-streaming siaran: Android dan iOS](#). Lihat juga [Catatan Rilis](#).

November 13, 2024

Filter Kamera Pihak Ketiga	Banyak perubahan dalam Menggunakan Snap dengan IVS Broadcast SDK > Web .	November 12, 2024
Siaran SDK: Web 1.18.0	<p>Nomor versi dan tautan artefak yang diperbarui dalam panduan SDK low-latency-streaming siaran: Web. Lihat juga Catatan Rilis.</p> <p>Kami menambahkan bagian baru, Dapatkan Informasi Peningkatan Tambahan (SEI), ke Panduan SDK.</p>	November 12, 2024
RTMP	Dalam “Buat Konfigurasi Ingest,” contoh yang diperbarui (ditambahkan -- ingest-protocol).	November 7, 2024
Siaran SDK: Web 1.17.0	<p>Nomor versi dan tautan artefak yang diperbarui di halaman arahan dokumentasi IVS dan dalam panduan SDK low-latency-streaming siaran: Web. Lihat juga Catatan Rilis.</p>	Oktober 10, 2024
Siaran SDK: Android 1.23.0, iOS 1.23.0	<p>Nomor versi dan tautan artefak yang diperbarui di halaman arahan dokumentasi IVS dan dalam panduan SDK real-time-streaming siaran: Android dan iOS. Lihat juga Catatan Rilis.</p> <p>Untuk Android, kami menambahkan Menggunakan SDK dengan Simbol Debug.</p>	Oktober 10, 2024

Service Quotas	Kami menambahkan kuota untuk “Peserta mempublikasikan bitrate.”	September 25, 2024
Memantau Streaming Waktu Nyata IVS	Menambahkan PublishFrameRate CloudWatch metrik.	September 13, 2024
Siaran SDK: Web 1.16.0	Nomor versi dan tautan artefak yang diperbarui di halaman arahan dokumentasi IVS dan dalam panduan SDK low-latency-streaming siaran: Web. Lihat juga Catatan Rilis .	September 11, 2024
Siaran SDK: Android 1.22.0, iOS 1.22.0	Nomor versi dan tautan artefak yang diperbarui di halaman arahan dokumentasi IVS dan dalam panduan SDK real-time-streaming siaran: Android dan iOS. Lihat juga Catatan Rilis . Kami juga memperbarui bagian, “Memulai> Instal Perpustakaan” di panduan SDK siaran Android.	September 11, 2024

[Tertelan RTMP](#)

Kami menambahkan halaman [IVS Stream Ingest](#). Di bawah ini ada dua halaman, RTMP (baru) dan WHIP.

September 9, 2024

Dalam [Menggunakan EventBridge dengan IVS Real-Time Streaming](#), kami menambahkan acara Pembaruan Tahap IVS, Kesalahan Publikasikan Peserta.

Di [Service Quotas](#), kami menambahkan nilai TPS untuk lima operasi API baru dan 1 kuota baru (dalam “IngestConfiguration Kuota Lainnya”).

Perubahan API dijelaskan dalam tabel [Referensi API](#).

[Optimasi Streaming Waktu Nyata](#)

Membuat berbagai pembaruan terkait simulcast dan menambahkan “Resolusi Lapisan.”

Agustus 22, 2024

[Terbitkan/berlangganan dalam konsol](#)

Dalam Memulai Streaming Waktu Nyata IVS, kami menambahkan penerbitan dalam konsol dan berlangganan [Publikasikan dan Berlangganan Video](#).

Agustus 19, 2024

[Siaran SDK: Web 1.15.0](#)

[Nomor versi dan tautan artefak yang diperbarui di halaman arahan dokumentasi IVS dan dalam panduan SDK low-latency-streaming siaran: Web.](#)

Lihat juga [Catatan Rilis](#).

Kami juga menambahkan bagian baru, [Konfigurasi untuk Berlangganan Peserta](#), ke Panduan SDK Siaran Web.

Dalam Pengoptimalan Streaming, kami menambahkan bagian baru, [Mengubah Buffer Jitter Pelanggan](#). MinDelay Ini termasuk informasi tentang Siaran Web, Android, dan iOS SDKs.

Agustus 15, 2024

[Siaran SDK: Android 1.21.0, iOS 1.21.0](#)

[Nomor versi dan tautan artefak yang diperbarui di halaman arahan dokumentasi IVS dan dalam panduan SDK real-time-streaming siaran: Android dan iOS.](#) Lihat juga [Catatan Rilis](#).

Kami juga menambahkan bagian baru, “Konfigurasi untuk Berlangganan Peserta,” ke Panduan SDK Siaran [Android](#) dan [iOS](#).

Agustus 15, 2024

Klarifikasi rekaman	Menambahkan catatan tentang penggunaan bucket S3 yang ada, dalam Perekaman Peserta Individu (dalam 1: Buat Bucket S3) dan Perekaman Komposit (dalam Prasyarat , Langkah 3). Pengaturan Kepemilikan Objek harus diberlakukan oleh pemilik Bucket atau pemilik Bucket lebih disukai.	Agustus 13, 2024
SDK Siaran: Web 1.14.0	Nomor versi dan tautan artefak yang diperbarui di halaman arahan dokumentasi IVS dan dalam panduan SDK low-latency-streaming siaran: Web. Lihat juga Catatan Rilis .	Juli 18, 2024
Siaran SDK: Android 1.20.0, iOS 1.20.0	Nomor versi dan tautan artefak yang diperbarui di halaman arahan dokumentasi IVS dan dalam panduan SDK real-time-streaming siaran: Android dan iOS. Lihat juga Catatan Rilis .	Juli 18, 2024
Memulai dengan Streaming Waktu Nyata	Menambahkan informasi tentang atribut ke “Distribusikan Token Peserta,” di “Skema Token: Payload” dan “Membuat Token dengan API Streaming Real-Time IVS.”	Juli 12, 2024
Service Quotas	Meningkatkan jumlah maksimum pelanggan panggung dari 10.000 menjadi 25.000.	27 Juni 2024

[Hasilkan Token Peserta dengan Pasangan Kunci](#)

Dalam Memulai dengan IVS Real-Time Streaming, kami memperbarui [Mendistribusikan Token Peserta](#) untuk menjelaskan dua cara untuk menghasilkan token (API dan key pair) dan kami menambahkan “Membuat Token dengan Pasangan Kunci.”

Juni 26, 2024

[Rekaman peserta individu](#)

Menambahkan bagian dokumentasi baru tentang [Perekaman](#), dengan sub-dokumen tentang [Perekaman Peserta Individu \(baru\) dan Rekaman Komposit](#) (sudah ada sebelumnya). Kami juga menambahkan acara dan contoh Perubahan Status Rekaman Peserta ke [Menggunakan EventBridge dengan IVS](#).

Juni 20, 2024

Perubahan API dijelaskan dalam tabel [Referensi API](#).

[Service Quotas](#)

Peningkatan kuota Tahapan dari 100 menjadi 1.000.

Juni 14, 2024

Siaran SDK: Web 1.13.0	Nomor versi dan tautan artefak yang diperbarui di halaman arahan dokumentasi IVS dan dalam panduan SDK low-latency-streaming siaran: Web. Lihat juga Catatan Rilis .	Juni 13, 2024
	Dalam panduan ini, kami memperbarui informasi di Penanganan Kesalahan untuk acara ERROR panggung baru.	
Siaran SDK: Android 1.19.0, iOS 1.19.0	Nomor versi dan tautan artefak yang diperbarui di halaman arahan dokumentasi IVS dan dalam panduan SDK real-time-streaming siaran: Android dan iOS. Lihat juga Catatan Rilis .	Juni 13, 2024
Siaran SDK: Web 1.12.0	Nomor versi dan tautan artefak yang diperbarui di halaman arahan dokumentasi IVS dan dalam panduan SDK low-latency-streaming siaran: Web. Lihat juga Catatan Rilis .	Mei 20, 2024
	Dalam panduan ini, kami memperbarui informasi dalam Menangani Masalah Jaringan tentang status koneksi tahapERRORRED.	
Optimasi Streaming Waktu Nyata	Dalam Default Layers, Quality, dan Framerates , mengubah bitrate maks untuk Mobile, Low Layer, dari 150.000 menjadi 100.000 bps.	16 Mei 2024

Siaran SDK: Android 1.18.0, iOS 1.18.0	Nomor versi dan tautan artefak yang diperbarui di halaman arahan dokumentasi IVS dan dalam panduan SDK real-time-streaming siaran: Android dan iOS. Lihat juga Catatan Rilis .	16 Mei 2024
Siaran SDK: Web 1.11.0	Nomor versi dan tautan artefak yang diperbarui di halaman arahan dokumentasi IVS dan dalam panduan SDK real-time-streaming siaran: Web. Lihat juga Catatan Rilis .	6 Mei 2024
Siaran SDK: Web 1.10.1	Nomor versi dan tautan artefak yang diperbarui di halaman arahan dokumentasi IVS dan dalam panduan SDK real-time-streaming siaran: Web. Lihat juga Catatan Rilis .	April 30, 2024
Siaran SDK: Android 1.15.2, iOS 1.15.2	Nomor versi dan tautan artefak yang diperbarui di halaman arahan dokumentasi IVS dan dalam panduan SDK real-time-streaming siaran: Android dan iOS. Lihat juga Catatan Rilis .	April 30, 2024
Broadcast SDK: Panduan iOS	Dalam Publish a Media Stream , kami memperbarui contoh kode.	April 26, 2024

[Siaran SDK: Android 1.17.0, iOS 1.17.0](#)

[Nomor versi dan tautan artefak yang diperbarui untuk rilis baru, dalam panduan SDK real-time-streaming siaran: Android dan iOS.](#) 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.

April 22, 2024

[Komposisi sisi server](#)

Di [SSC](#), membuat berbagai perubahan, terutama di “Layout,” untuk menjelaskan PiP dan tata letak grid.

26 Maret 2024

Dalam Panduan SDK Broadcast Web, tambahkan Dukungan Rendering [Sisi Server](#).

[OBS dan WHIP Support](#)

Menambahkan catatan tentang masalah kualitas (seperti pembekuan video intermiten) yang dapat terjadi dengan WHIP di OBS.

Maret 22, 2024

[Siaran SDK: Android 1.16.0, iOS 1.16.0, Web 1.10.0](#)

[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.

Maret 21, 2024

[Siaran SDK: Android 1.15.1, iOS 1.15.1](#)

[Nomor versi dan tautan artefak yang diperbarui untuk rilis baru, dalam panduan SDK real-time-streaming siaran: Android dan iOS.](#) 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.

Maret 13, 2024

[Siaran SDK: Mode Audio Seluler](#)

Di “Preset Mode Audio,” menambahkan informasi tentang kategori preset Volume Rocker dan masalah iOS yang diketahui dengan preset Obrolan Video. Di “Kasus Penggunaan Lanjutan,” menambahkan catatan tentang menghindari konfigurasi yang salah, dan menambahkan bagian pada “Pembatalan Echo iOS” dan “Sumber Audio Kustom iOS.”

Maret 1, 2024

[Siaran SDK: Android 1.15.0, iOS 1.15.0, Web 1.9.0](#)

[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.

Februari 22, 2024

[OBS dan WHIP Support](#)

Ditambahkan halaman baru.

Februari 6, 2024

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.

[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.

Februari 1, 2024

Untuk Panduan Android, kami menambahkan Masalah Diketahui baru (ukuran video kurang dari 176x176).

Untuk Panduan Web, kami menambahkan Masalah Baru yang Diketahui. Solusinya adalah membatasi resolusi video ke 720p saat memanggil atau `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

[Glosarium IVS](#)

Memperlengkap glosarium, yang mencakup istilah-istilah waktu nyata, latensi rendah, dan obrolan IVS.

20 Desember 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, IVSRead OnlyAccess dan IVSFull Access. 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.

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](#)

[\(Streaming Latensi Rendah\).](#)

[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.

[SDK siaran IVS](#)

Dalam [ikhtisar Broadcast SDK](#), 9 November 2023

kami memperbarui Persyaratan Platform > Platform Asli untuk memperjelas versi SDK mana yang didukung dan kami menambahkan “Browser Seluler (iOS dan Android).”

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	<p>Nomor versi dan tautan artefak yang diperbarui untuk rilis baru, dalam panduan SDK real-time-streaming siaran: Web.</p> <p>Halaman arahan dokumentasi Amazon IVS menunjuk ke versi Referensi SDK Siaran saat ini.</p> <p>Lihat juga Catatan Rilis Amazon IVS untuk rilis ini.</p> <p>Dalam Panduan Web, di “Ambil MediaStream dari Perangkat,” kami juga menghapus dua max baris; praktik terbaik adalah menentukan saja <code>ideal</code>.</p> <p>Dalam Optimasi Streaming Waktu Nyata, kami menambahkan bagian baru, Mengoptimalkan Bitrate Audio dan Dukungan Stereo.</p>	16 Oktober 2023

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	<p>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.</p> <p>Halaman arahan dokumentasi Amazon IVS menunjuk ke versi Referensi SDK Siaran saat ini.</p> <p>Lihat juga Catatan Rilis Amazon IVS untuk rilis ini.</p>	12 Oktober 2023
Siaran SDK: Web 1.5.2	<p>Nomor versi dan tautan artefak yang diperbarui untuk rilis baru, dalam panduan SDK real-time-streaming siaran: Web.</p> <p>Halaman arahan dokumentasi Amazon IVS menunjuk ke versi Referensi SDK Siaran saat ini.</p> <p>Lihat juga Catatan Rilis Amazon IVS untuk rilis ini.</p>	14 September 2023
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	<p>Nomor versi dan tautan artefak yang diperbarui untuk rilis baru, dalam panduan SDK real-time-streaming siaran: Web, Android, dan iOS.</p> <p>Pada halaman arahan dokumentasi Amazon IVS, memperbarui tautan Referensi SDK siaran untuk menunjuk ke versi baru.</p> <p>Lihat juga Catatan Rilis Amazon IVS untuk rilis ini.</p>	23 Agustus 2023

[Peluncuran streaming waktu nyata](#)

Rilis ini disertai perubahan besar pada dokumentasi. 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	Date
Perbarui StartParticipantReplication operasi	Di StartParticipantReplication, ubah nilai maksimum untuk <code>reconnectWindowSeconds</code> bidang dari 60 menjadi 300.	Maret 16, 2026
Pertukaran token peserta	<p>Menambahkan <code>ExchangedParticipantToken</code> objek.</p> <p>Ditambahkan <code>newToken</code> dan <code>previousToken</code> bidang ke objek Event.</p> <p>Ditambahkan <code>TOKEN_EXCHANGED</code> sebagai nilai yang valid untuk <code>name</code> bidang di objek Event.</p> <p>Perubahan Panduan Pengguna dijelaskan dalam tabel Panduan Pengguna.</p>	Desember 9, 2025
Pemesanan peserta khusus SSC	<p>Memodifikasi <code>GridConfiguration</code> dan tipe <code>PipConfiguration</code> data (menambahkan <code>participantOrderAttribute</code> bidang), yang pada gilirannya mempengaruhi objek Komposisi. Hal ini mempengaruhi <code>StartComposition</code> permintaan dan respon dan <code>GetComposition</code> respon.</p> <p>Perubahan Panduan Pengguna dijelaskan dalam tabel Panduan Pengguna.</p>	September 16, 2025
Replikasi peserta	<p>Rilis awal dari fungsi baru ini. Lihat perubahan dokumentasi ini:</p> <ul style="list-style-type: none"> Menambahkan terminologi khusus untuk replikasi peserta dalam “Konsep Kunci.” Menambahkan tiga operasi baru: <code>ListParticipantReplicas</code>, <code>StartParticipantReplication</code>, <code>StopParticipantReplication</code>. 	29 Mei 2025

Perubahan API	Deskripsi	Date
	<ul style="list-style-type: none"> • Ditambahkan satu objek baru, ParticipantReplica. • Menambahkan recordParticipantReplicas bidang ke AutoParticipantRecordingConfiguration objek. Hal ini mempengaruhi CreateStage permintaan dan respon, GetStage respon, dan UpdateStage permintaan dan respon. • Ditambahkan tiga bidang ke objek Event:destinationStageArn ,destinationSessionId ,replica. Ini mempengaruhi ListParticipantEvents respons. • Menambahkan empat bidang ke Peserta dan ParticipantSummary objek:replicationState ,replicationType ,sourceStageArn ,sourceSessionId . Ini mempengaruhi GetParticipant dan ListParticipants tanggapan. • Dalam Acara, menambahkan dua nilai yang valid ke name bidang:REPLICATION_STARTED ,REPLICATION_STOPPED . <p>Perubahan Panduan Pengguna dijelaskan dalam tabel Panduan Pengguna.</p>	

Perubahan API	Deskripsi	Date
Durasi Segmen Target	<p>Memodifikasi <code>AutoParticipantRecordingConfiguration</code> objek (menambahkan <code>hlsConfiguration</code> bidang); ini pada gilirannya mempengaruhi objek Stage. Hal ini mempengaruhi <code>CreateStage</code> permintaan dan respon, <code>GetStage</code> respon, dan <code>UpdateStage</code> permintaan dan respon.</p> <p>Memodifikasi <code>RecordingConfiguration</code> objek (menambahkan <code>hlsConfiguration</code> bidang). Hal ini mempengaruhi <code>GetComposition</code> respon dan <code>StartComposition</code> permintaan dan respon.</p> <p>Menambahkan dua objek: <code>CompositionRecordingHlsConfiguration</code> dan <code>ParticipantRecordingHlsConfiguration</code>.</p>	Maret 13, 2025
Jahitan perekaman peserta individu	<p>Memodifikasi <code>AutoParticipantRecordingConfiguration</code> objek (menambahkan <code>recordingReconnectWindowSeconds</code> bidang); ini pada gilirannya mempengaruhi objek Stage. Hal ini mempengaruhi <code>CreateStage</code> permintaan dan respon, <code>GetStage</code> respon, dan <code>UpdateStage</code> permintaan dan respon.</p> <p>Memperbarui <code>desripsiParticipant.recordingS3Prefix</code>.</p>	6 Maret 2025

Perubahan API	Deskripsi	Date
Konfigurasi thumbnail waktu nyata	<p>Memodifikasi DestinationConfiguration objek S3: ditambahkan thumbnailConfigurations . Hal ini mempengaruhi GetComposition respon dan StartComposition permintaan dan respon.</p> <p>Memodifikasi AutoParticipantRecordingConfiguration objek: ditambahkan thumbnailConfiguration dan ditambahkan NONE sebagai nilai yang valid untuk mediaTypes . Hal ini mempengaruhi CreateStage permintaan dan respon, GetStage respon, dan UpdateStage permintaan dan respon.</p> <p>Menambahkan dua objek: CompositionThumbnailConfiguration dan ParticipantThumbnailConfiguration.</p>	Desember 10, 2024
Perbarui objek Acara dan Video	<p>Dalam objek Event, kami menambahkan nilai yang lebih valid untuk errorCode .</p> <p>Dalam objek Video, kami mengklarifikasi itu height dan width harus berupa angka genap.</p>	Oktober 2, 2024

Perubahan API	Deskripsi	Date
Tertelan RTMP	<p>Kami menambahkan dua objek: <code>IngestConfiguration</code> dan <code>IngestConfigurationSummary</code>. Kami menambahkan lima <code>IngestConfiguration</code> titik akhir (Buat, Hapus, Dapatkan, Daftar, dan Perbarui).</p> <p>Kami memperbarui <code>DeleteStage</code> (deskripsi operasi) dan <code>DisconnectParticipant</code> (deskripsi operasi dan <code>participantId</code>).</p> <p>Kami memodifikasi objek Peserta (menambahkan <code>protocol</code> bidang); yang memengaruhi <code>GetParticipant</code> respons.</p> <p>Kami memodifikasi <code>StageEndpoints</code> objek (menambahkan <code>rtmp</code> dan <code>rtmps</code> bidang); yang mempengaruhi <code>CreateStage</code>, <code>GetStage</code>, dan <code>UpdateStage</code> tanggapan. Kami juga memperbarui deskripsi objek ini (menambahkan rekomendasi <code>cacheing</code>).</p>	September 9, 2024
Hasilkan Token Peserta dengan Pasangan Kunci	<p>Kami menambahkan tiga objek (<code>PublicKey</code>, <code>PublicKeySummary</code>, <code>StageEndpoints</code>) dan empat titik akhir: (<code>DeletePublicKey</code>, <code>GetPublicKey</code>, <code>ImportPublicKey</code>, <code>ListPublicKeys</code>). Kami memodifikasi objek <code>Stage</code> (menambahkan <code>endpoints</code> bidang); yang mempengaruhi <code>CreateStage</code>, <code>GetStage</code>, dan <code>UpdateStage</code> tanggapan.</p>	Juni 26, 2024
Rekaman peserta individu	<p>Kami menambahkan satu objek (<code>AutoParticipantRecordingConfiguration</code>) dan memodifikasi tiga objek (Peserta <code>ParticipantSummary</code>, <code>Tahap</code>). Ini memengaruhi lima titik akhir: <code>CreateStage</code> permintaan dan respons, <code>GetParticipant</code> respons, <code>GetStage</code> respons, <code>ListParticipants</code> permintaan dan respons, serta <code>UpdateStage</code> permintaan dan respons.</p>	Juni 20, 2024

Perubahan API	Deskripsi	Date
Hapus svcs dari pola ARN	Pola ARN yang ditentukan [is]vs diperbarui untuk ditentukan. <code>ivs</code> Ini mempengaruhi ketiga titik akhir Tag dan <code>ChannelDestinationConfiguration\$channelArn</code> bidang.	April 25, 2024
Pembaruan komposisi sisi server	Kami menambahkan satu objek: <code>PipConfiguration</code> . Kami memodifikasi dua objek (<code>LayoutConfiguration</code> , <code>GridConfiguration</code>). Hal ini mempengaruhi <code>GetComposition</code> respon dan <code>StartComposition</code> permintaan dan respon.	Maret 13, 2024
Rekaman komposit	Kami menambahkan 4 <code>StorageConfiguration</code> titik akhir dan 7 objek (<code>DestinationDetail</code> , <code>S3RecordingConfiguration</code> , <code>S3DetailDestinationConfiguration</code> , <code>S3</code> ,). <code>StorageConfiguration</code> <code>StorageConfiguration</code> <code>StorageConfigurationSummary</code> Kami memodifikasi 3 objek (<code>Komposisi</code> , <code>Tujuan</code> , <code>DestinationConfiguration</code>). Hal ini mempengaruhi <code>GetComposition</code> respon dan <code>StartComposition</code> permintaan dan respon.	16 November 2023
Komposisi di sisi server	Kami menambahkan 8 Komposisi dan <code>EncoderConfiguration</code> titik akhir dan 11 objek (<code>ChannelDestinationConfiguration</code> , <code>Komposisi</code> , <code>CompositionSummary</code> , <code>Tujuan</code> , <code>DestinationConfiguration</code> , <code>Destinati onSummary</code> , <code>EncoderConfiguration</code> , <code>EncoderCo nfigurationSummary</code> , <code>GridConfiguration</code> <code>LayoutCon figuration</code> , dan <code>Video</code>).	16 November 2023
Stage Health: Data Peserta Baru	Menambahkan enam bidang ke objek Peserta : <code>browserName</code> , <code>browserVersion</code> , <code>ispName</code> , <code>osName</code> , <code>osVersion</code> , dan <code>sdkVersion</code> . Ini mempengaruhi <code>GetParticipant</code> respons.	12 Oktober 2023

Perubahan API	Deskripsi	Date
Token Peserta	Menambahkan catatan Penting tentang tidak membangun fungsionalitas berdasarkan format token saat ini.	1 September 2023
Peluncuran Streaming Waktu Nyata IVS	<p>Rilis ini disertai perubahan besar pada dokumentasi. 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.</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>	Agustus 7, 2023

Catatan Rilis IVS | Streaming Waktu Nyata

Dokumen ini berisi semua catatan rilis Amazon IVS Real-Time Streaming, terbaru pertama, diatur berdasarkan tanggal rilis.

Maret 12, 2026

SDK Siaran IVS: Web 1.33.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
Siaran Web SDK 1.33.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-referensi</p> <ul style="list-style-type: none">• Menerapkan <code>exchangeToken</code> metode untuk pertukaran token real-time.• Menerapkan acara <code>STAGE_PARTICIPANT_METADATA_CHANGED</code>, yang diaktifkan saat perubahan setelah pertukaran token. <code>attributes</code> and/or <code>userId</code>• Ditambahkan <code>encoderImplementation</code> bidang pada permintaan <code>requestQualityStats()</code> metode aliran tahap lokal.

Maret 12, 2026

SDK Siaran Amazon IVS: Android 1.40.0, iOS 1.40.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Android 1.40.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.40.0/android/</p>

Platform	Unduhan dan Perubahan
	<ul style="list-style-type: none"> • Pesan kesalahan yang disempurnakan seputar kegagalan validasi sertifikat TLS dan kode enum kesalahan yang diperluas.
SDK Siaran iOS 1.40.0	<p>Unduh untuk streaming waktu nyata: https://broadcast.live-video.net/1.40.0/AmazonIVSBroadcast -Stages.xcFramework.zip</p> <p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.40.0/ios/</p> <ul style="list-style-type: none"> • Pesan kesalahan yang disempurnakan seputar kegagalan validasi sertifikat TLS dan kode enum kesalahan yang diperluas.

Ukuran SDK Siaran: Android

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64-v8a	5,823 MB	14.139 MB
armeabi-v7a	5,046 MB	9,798 MB
x86_64	5,935 MB	14.702 MB
x86	6.190 MB	15.265 MB

Ukuran SDK Siaran: iOS

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64	3,939 MB	8,013 MB

Februari 13, 2026

SDK Siaran Amazon IVS: Android 1.39.0, iOS 1.39.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Android 1.39.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.39.0/android/</p> <ul style="list-style-type: none">• Perbaiki bug kecil untuk headset Bluetooth terhubung kembali menggunakan mode audio STUDIO.• Alat build Android inti yang diperbarui dan versi NDK.• Memperbaiki kebuntuan langka saat menghentikan a. MixedImageDevice
SDK Siaran iOS 1.39.0	<p>Unduh untuk streaming waktu nyata: https://broadcast.live-video.net/1.39.0/AmazonIVSBroadcast -Stages.xcFramework.zip</p> <p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.39.0/ios/</p> <ul style="list-style-type: none">• Diperbarui Xcode ke versi 26.2.• Efektif dengan rilis ini, IVS SDKs tidak lagi didistribusikan melalui CocoaPods. <p>CocoaPods mengumumkan penghentiannya pada tahun 2024 dan akan memasuki status hanya-baca akhir tahun ini. Swift Package Manager (SPM) menggantikan CocoaPods solusi manajemen ketergantungan Apple yang didukung dan merupakan cara standar untuk berintegrasi dalam proyek Xcode modern. SDKs</p>

Platform	Unduhan dan Perubahan
	<p>Kami menyarankan Anda bermigrasi ke SPM atau mengintegrasikan kerangka kerja SDK IVS langsung ke proyek Anda. IVS SDKs didukung penuh melalui kedua pendekatan.</p> <p>Perubahan dokumentasi terkait dibuat di:</p> <ul style="list-style-type: none"> • Memulai Streaming Waktu Nyata IVS - “Langkah 4: Integrasikan SDK Siaran IVS > “iOS” • Panduan SDK Siaran iOS — “Instal Perpustakaan”

Ukuran SDK Siaran: Android

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64-v8a	5,788 MB	14.059 MB
armeabi-v7a	5.016 MB	9,740 MB
x86_64	5,898 MB	14.615 MB
x86	6.154 MB	15.184 MB

Ukuran SDK Siaran: iOS

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64	3,932 MB	7,996 MB

Februari 12, 2026

SDK Siaran IVS: Web 1.32.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
Siaran Web SDK 1.32.0	Dokumentasi referensi: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-referensi <ul style="list-style-type: none"> • Perbaiki bug dan peningkatan stabilitas.

Januari 13, 2026

SDK Siaran Amazon IVS: Android 1.38.0, iOS 1.38.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Android 1.38.0	Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.38.0/android/ <ul style="list-style-type: none"> • Fungsi prioritas lapisan - Menambahkan an Priority enum StageVideoConfiguration.Layer dengan nilai:VERY_LOW,,LOW. MEDIUM HIGH Ini akan menentukan lapisan mana yang dijatuhkan pertama kali di bawah kendala bandwidth jaringan. • Koneksi ulang tahap lebih cepat setelah konektivitas jaringan dipulihkan. • Mengubah kode yang terkait dengan beberapa kesalahan. Lihat Panduan Migrasi Kesalahan SDK Siaran Seluler di bawah ini.

Platform	Unduhan dan Perubahan
SDK Siaran iOS 1.38.0	<p>Unduh untuk streaming waktu nyata: https://broadcast.live-video.net/1.38.0/AmazonIVSBroadcast -Stages.xcFramework.zip</p> <p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.38.0/ios/</p> <ul style="list-style-type: none"> • Fungsi prioritas lapisan - Menambahkan <code>IVSLocalStageStreamLayerPriority</code> enum dengan nilai: <code>VeryLow</code>, <code>Low</code>, <code>Medium</code>, <code>High</code>. Ini akan menentukan lapisan mana yang dijatuhkan pertama kali di bawah kendala bandwidth jaringan. • Koneksi ulang tahap lebih cepat setelah konektivitas jaringan dipulihkan. • Mengubah kode yang terkait dengan beberapa kesalahan. Lihat Panduan Migrasi Kesalahan SDK Siaran Seluler di bawah ini.

Ukuran SDK Siaran: Android

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64-v8a	5,795 MB	14.070 MB
armeabi-v7a	5.021 MB	9,746 MB
x86_64	5,904 MB	14.630 MB
x86	6.161 MB	15.198 MB

Ukuran SDK Siaran: iOS

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64	3,609 MB	8,078 MB

Panduan Migrasi Kesalahan SDK Siaran Seluler

Dalam versi 1.38.0 dari siaran iOS dan Android SDKs, kode yang terkait dengan beberapa kesalahan telah berubah. Sebelumnya, tidak ada properti tunggal yang dapat digunakan untuk mengidentifikasi secara unik kesalahan yang dipancarkan dari SDKs. Sebagai gantinya, untuk memahami apa arti kesalahan, kombinasi properti berikut perlu diperiksa:

Android	iOS
<code>BroadcastException.getCode()</code>	<code>NSError.code</code>
<code>BroadcastException.getUid()</code>	<code>NSError.userInfo[IVSBroadcastUidDescriptionErrorKey]</code>
<code>BroadcastException.getError()</code>	<code>NSError.userInfo[IVSBroadcastResultDescriptionErrorKey]</code>
<code>BroadcastException.getSource()</code>	<code>NSError.userInfo[IVSBroadcastSourceDescriptionErrorKey]</code>
<code>BroadcastException.getDetail()</code>	<code>NSError.userInfo[NSLocalizedStringDescriptionKey]</code>

Dengan versi 1.38.0 dan yang lebih besar, `BroadcastException.getCode()` (Android) dan `(NSError.code)` (iOS) mengembalikan ID unik yang dapat dicari di enum publik `BroadcastErrorCode` (Android) dan `(IVSBroadcastErrorCode)` (iOS).

Selain membuat code ID unik untuk semua kesalahan, bidang tambahan ditambahkan: `BroadcastException.getPlatformCode()` (Android) dan `NSError.userInfo[IVSBroadcastPlatformCodeDescriptionErrorKey]` (iOS). Jika kesalahan disebabkan oleh platform yang mendasarinya (seperti kesalahan jaringan atau kesalahan

encode video atau decode), bidang ini bukan nol dan dapat digunakan untuk mengumpulkan informasi tambahan dari dokumentasi platform.

Migrasi dari SDK 1.37.0 dan Sebelumnya

Untuk membuat setiap kesalahan sesuai dengan strategi baru, beberapa kesalahan yang ada harus mengubah nilainya. Di bawah ini adalah panduan untuk memetakan logika yang ada ke logika baru:

- Kesalahan apa pun yang code bukan nol akan mempertahankan nilai kode yang sama; namun, mereferensikan kode melalui konstanta enum baru dapat meningkatkan kejelasan. Misalnya, membandingkan kesalahan dengan lebih `BroadcastErrorCode.Broadcast.LatencyThresholdReached` jelas daripada membandingkannya `20401`.
- Kesalahan apa pun yang UID memiliki nilai (yaitu tidak ada `-1` di Android atau `"-1"` di iOS) sekarang akan memiliki code bidang yang disetel ke UID nilai yang ada. Jika Anda memiliki kondisional yang membandingkan UID bidang, Anda dapat menyimpan konstanta tetapi membandingkannya dengan code bidang di masa depan.
- Beberapa kesalahan lama tidak mengandung nilai code atau UID nilai. Ini biasanya dicocokkan berdasarkan kesalahan message (Android) atau `description` (iOS), yang bukan merupakan cara yang dapat diandalkan untuk mengidentifikasi kesalahan karena sifat dinamis dari pesan kesalahan. Karena kesalahan ini tidak memiliki karakteristik identifikasi yang unik, one-to-one pemetaan tidak dapat disediakan. Namun, sebagian besar kesalahan menyimpan deskripsi yang sama, sehingga dimungkinkan untuk terus menggunakan logika pencocokan yang sama sambil juga mengumpulkan dan melaporkan code nilai baru untuk rilis aplikasi future.

Sebagai contoh konkret, pemeriksaan kesalahan pada tabel berikut harus dimigrasikan sebagai berikut:

Sebelum	Setelah
<code>error.code == 20401</code>	<code>error.code == BroadcastErrorCode .Broadcast.LatencyThreshold Reached</code> Tidak ada perubahan, tetapi lebih suka perbandingan dengan nilai enum.

Sebelum	Setelah
<code>error.uid == 207</code>	<code>error.code == BroadcastErrorCode .Net.SocketRemoteHangup</code> Bandingkan dengan code bukannyauid.
<code>error.message.contains("Ice ConnectionFailed")</code>	<code>error.code == BroadcastErrorCode .RealTime.PeerConnectionIce ConnectionFailed</code> Jangan membandingkan dengan message (<code>atausource</code> , <code>atauresult/detail</code>). Sebagai gantinya, temukan kode enum yang sesuai untuk dibandingkan.

Bagian terpenting dari kesalahan masih `BroadcastException.getPlatformCode()` (Android) dan `NSError.userInfo[IVSBroadcastPlatformCodeDescriptionErrorKey]` (iOS), tetapi dalam versi 1.38.0 dan seterusnya, code bidang ini secara unik mengidentifikasi kesalahan dan memungkinkan pencarian langsung nama kesalahan dan deskripsi di enum (`BroadcastErrorCodeAndroid`) dan (iOS). `IVSBroadcastErrorCode` Akibatnya, bidang lain seperti `UID`, `source`, dan `detail` boleh digunakan dalam logika pencarian; mereka hanya ada sebagai informasi tambahan.

Desember 11, 2025

SDK Siaran Amazon IVS: Android 1.37.1 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Android 1.37.1	Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.37.1/android/ <ul style="list-style-type: none"> Memperbaiki masalah yang terkait dengan pembongkaran pratinjau peserta.

Ukuran SDK Siaran: Android

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64-v8a	5,754 MB	13.965 MB
armeabi-v7a	4,991 MB	9,683 MB
x86_64	5,858 MB	14.529 MB
x86	6,128 MB	15.120 MB

Desember 9, 2025

Pertukaran Token Peserta

Dukungan baru untuk pertukaran token peserta memungkinkan Anda untuk meningkatkan atau menurunkan kemampuan token dan memperbarui atribut token dalam SDK klien IVS tanpa memaksa klien untuk memutuskan dan menyambung kembali. Ini berguna untuk skenario seperti co-hosting, di mana peserta dapat memulai dengan kemampuan berlangganan saja dan kemudian membutuhkan kemampuan publikasi.

Lihat halaman baru di [Token Exchange](#).

Desember 5, 2025

SDK Siaran IVS: Web 1.31.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
Siaran Web SDK 1.31.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-referensi</p> <ul style="list-style-type: none"> Perbaikan bug dan peningkatan stabilitas.

Desember 5, 2025

SDK Siaran Amazon IVS: Android 1.37.0, iOS 1.37.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Android 1.37.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.37.0/android/</p> <ul style="list-style-type: none"> • Perbaiki bug dan peningkatan stabilitas. • Menambahkan dukungan untuk pertukaran token peserta.
SDK Siaran iOS 1.37.0	<p>Unduh untuk streaming waktu nyata: https://broadcast.live-video.net/1.37.0/AmazonIVSBroadcast-Stages.xcFramework.zip</p> <p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.37.0/ios/</p> <ul style="list-style-type: none"> • Perbaiki bug dan peningkatan stabilitas. • Menambahkan dukungan untuk pertukaran token peserta.

Ukuran SDK Siaran: Android

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64-v8a	5,753 MB	13.961 MB
armeabi-v7a	4,990 MB	9,680 MB
x86_64	5,857 MB	14.525 MB
x86	6.127 MB	15.116 MB

Ukuran SDK Siaran: iOS

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64	3,588 MB	8,028 MB

November 7, 2025

Sinkronisasi Rekaman Peserta Individu

Dukungan baru untuk EXT-X-PROGRAM-DATE-TIME tag dalam daftar putar HLS perekaman peserta individu memungkinkan sinkronisasi yang tepat dari beberapa rekaman peserta selama pasca-pemrosesan. Fitur ini menyediakan stempel waktu UTC akurat milidetik saat merekam titik awal dan diskontinuitas, memungkinkan Anda membuat komposisi yang disinkronkan (seperti side-by-side atau picture-in-picture tata letak) bahkan ketika peserta mengalami gangguan jaringan atau bergabung pada waktu yang berbeda. Untuk detailnya, lihat [Menyinkronkan Beberapa Rekaman Peserta dalam Rekaman Peserta Individu](#).

Oktober 30, 2025

SDK Siaran IVS: Web 1.30.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
Siaran Web SDK 1.30.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-referensi</p> <ul style="list-style-type: none">Perbaikan bug dan peningkatan stabilitas.

Oktober 30, 2025

SDK Siaran Amazon IVS: Android 1.36.0, iOS 1.36.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Android 1.36.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.36.0/android/</p> <ul style="list-style-type: none">• Peningkatan pemulihan kamera saat kembali ke latar depan setelah berada di latar belakang untuk jangka waktu yang lama.• Menambahkan <code>embedMessage</code> metode <code>ImageDevice</code> untuk mengaktifkan penyisipan muatan metadata ke dalam aliran video penerbitan. Lihat Menyematkan Pesan di Panduan SDK Siaran Android.
SDK Siaran iOS 1.36.0	<p>Unduh untuk streaming waktu nyata: https://broadcast.live-video.net/1.36.0/AmazonIVSBroadcast-Stages.xcFramework.zip</p> <p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.36.0/ios/</p> <ul style="list-style-type: none">• Menambahkan <code>embedMessage</code> metode <code>IVSImageDevice</code> untuk mengaktifkan penyisipan muatan metadata ke dalam aliran video penerbitan. Lihat Menyematkan Pesan di Panduan SDK Siaran iOS.

Ukuran SDK Siaran: Android

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64-v8a	5,736 MB	13.898 MB
armeabi-v7a	4,974 MB	9,638 MB
x86_64	5,839 MB	14.456 MB
x86	6,109 MB	15.047 MB

Ukuran SDK Siaran: iOS

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64	3,569 MB	7,962 MB

Oktober 14, 2025

Batas Waktu Nyata yang Diperbarui: Komposisi

Kami memperbarui kuota untuk “sumber daya Komposisi bersamaan maksimum per akun” dari 5 menjadi 20. Hal ini didokumentasikan dalam Service Quotas > [Other](#) Quotas.

Oktober 2, 2025

SDK Siaran IVS: Web 1.29.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
Siaran Web SDK 1.29.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-referensi</p> <ul style="list-style-type: none"> Perbaikan bug dan peningkatan stabilitas.

Oktober 2, 2025

SDK Siaran Amazon IVS: Android 1.35.0, iOS 1.35.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Android 1.35.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.35.0/android/</p> <ul style="list-style-type: none"> Perbaikan bug dan peningkatan stabilitas.
SDK Siaran iOS 1.35.0	<p>Unduh untuk streaming waktu nyata: https://broadcast.live-video.net/1.35.0/AmazonIVSBroadcast -Stages.xcFramework.zip</p> <p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.35.0/ios/</p> <ul style="list-style-type: none"> <code>IVSImageDevice.setOnFrameCallback</code> sekarang dapat disesuaikan dengan <code>DispatchQueue</code>, dan secara opsional dapat menyertakan yang <code>CVPixelBuffer</code> terkait dengan bingkai.

Ukuran SDK Siaran: Android

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64-v8a	5,730 MB	13.900 MB
armeabi-v7a	4,971 MB	9,639 MB
x86_64	5,835 MB	14.455 MB
x86	6,104 MB	15.041 MB

Ukuran SDK Siaran: iOS

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64	3,569 MB	7,963 MB

September 16, 2025

Komposisi Sisi Server Pemesanan Peserta Kustom

Dukungan baru untuk pemesanan peserta khusus untuk SSC memberikan kontrol terperinci atas pemosisian peserta di tata letak grid dan Picture-in-Picture (PiP). Lihat [Komposisi Sisi Server](#) (berbagai perubahan, termasuk menambahkan `participantOrderAttribute` dan “Pemesanan Peserta Kustom”) dan [Referensi API Streaming Waktu Nyata IVS](#) (ditambahkan `participantOrderAttribute` ke objek Komposisi).

September 11, 2025

SDK Siaran Amazon IVS: Android 1.34.0, iOS 1.34.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Android 1.34.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.34.0/android/</p> <ul style="list-style-type: none"> • Peningkatan CPU untuk mempublikasikan dan berlangganan transportasi media. • Menambahkan <code>packetsLost</code> ke <code>LocalVideoStats</code> dan <code>LocalAudioStats</code>.
SDK Siaran iOS 1.34.0	<p>Unduh untuk streaming waktu nyata: https://broadcast.live-video.net/1.34.0/AmazonIVSBroadcast-Stages.xcFramework.zip</p>

Platform	Unduhan dan Perubahan
	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.34.0/ios/</p> <ul style="list-style-type: none"> • Peningkatan CPU untuk mempublikasikan dan berlangganan transportasi media. • Menambahkan <code>packetsLost</code> ke <code>IVSLocalVideoStats</code> dan <code>IVSLocalAudioStats</code>. • Memperbaiki bug di mana perangkat tidak terlepas setelah meninggalkan panggung, yang dapat mengakibatkan indikator privasi tiba-tiba menyala.

Ukuran SDK Siaran: Android

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64-v8a	5,796 MB	14.089 MB
armeabi-v7a	5,036 MB	9,788 MB
x86_64	5,906 MB	14.653 MB
x86	6.174 MB	15,240 MB

Ukuran SDK Siaran: iOS

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64	3,594 MB	8,046 MB

September 10, 2025

Antarmuka VPC Endpoint

Dukungan baru untuk titik akhir VPC (Virtual Private Cloud) antarmuka memungkinkan Anda membuat koneksi pribadi yang aman antara VPC Amazon dan IVS Anda, untuk beban kerja yang memerlukan konsumsi video langsung yang aman. Ini membuat lalu lintas menelan IVS Anda di dalam jaringan AWS dan di luar internet publik. Endpoint VPC antarmuka didukung oleh AWS, teknologi AWS PrivateLink yang memungkinkan komunikasi pribadi antara layanan AWS, menggunakan antarmuka network elastis dengan pribadi di Amazon VPC IPs Anda. Lihat [Pencerapan Pribadi](#) di Panduan Pengguna Streaming Latensi Rendah IVS dan [Ingest Pribadi ke Tahapan](#) dalam Panduan Pengguna Streaming Waktu Nyata IVS.

September 4, 2025

SDK Siaran IVS: Web 1.28.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
Siaran Web SDK 1.28.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-referensi</p> <ul style="list-style-type: none">Bergabung dengan tahap yang telah dihapus, atau dengan token peserta yang terputus, sekarang melaporkan <code>STAGE_DELETED</code> atau <code>STAGE_DISCONNECTED</code> kesalahan, bukan. <code>TIMEOUT</code>Permintaan polling internal yang dioptimalkan terkait dengan simulcast.

Agustus 7, 2025

SDK Siaran IVS: Web 1.27.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Web 1.27.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-referensi</p> <ul style="list-style-type: none">• Ditambahkan <code>requestQualityStats</code> ke <code>RemoteStageStream</code>, yang kemudian mengekspos objek yang disederhanakan dari statistik video dan audio yang bersumber dari <code>requestRTCStats</code>• Pembaruan untuk memastikan bahwa status yang <code>RemoteStageStream</code> diredam dan status yang <code>mediaStreamTrack</code> diaktifkan selalu sinkron.

Agustus 7, 2025

SDK Siaran Amazon IVS: Android 1.33.0, iOS 1.33.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Android 1.33.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.33.0/android/</p> <ul style="list-style-type: none">• Metode baru untuk mengontrol obor perangkat:<ul style="list-style-type: none">• <code>CameraSource.Capabilities</code> mengimplementasikan <code>isTorchSupported</code>

Platform	Unduhan dan Perubahan
	<ul style="list-style-type: none"> • <code>CameraSource.Options.Builder</code> mengimplementasikan <code>setEnabledTorch</code> • SDK siaran Android memenuhi persyaratan kompatibilitas ukuran halaman 16 KB Google Play. (Catatan: Ini diterapkan pada SDK versi 1.23.0.)
SDK Siaran iOS 1.33.0	<p>Unduh untuk streaming waktu nyata: https://broadcast.live-video.net/1.33.0/AmazonIVSBroadcast-Stages.xcFramework.zip</p> <p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.33.0/ios/</p> <ul style="list-style-type: none"> • Metode baru untuk mengontrol obor perangkat: <code>IVSImageDevice</code> mengimplementasikan dua properti, <code>isTorchSupported</code> dan <code>torchEnabled</code>. Periksa apakah perangkat mendukung obor dengan <code>isTorchSupported</code>, dan kemudian beralih dengan pengaturan <code>torchEnabled</code>. • Menyelesaikan masalah di iOS 18.5+ dengan pasti VPNs yang dapat mengakibatkan batas waktu koneksi rekan. (Catatan: Ini diterapkan pada SDK versi 1.32.1.)

Ukuran SDK Siaran: Android

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64-v8a	5,689 MB	13.829 MB

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
armeabi-v7a	4,962 MB	9,649 MB
x86_64	5,806 MB	14.413 MB
x86	6,066 MB	14.983 MB

Ukuran SDK Siaran: iOS

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64	3,505 MB	7,828 MB

Juli 25, 2025

SDK Siaran Amazon IVS: Android 1.32.2 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Android 1.32.2	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.32.2/android/</p> <ul style="list-style-type: none"> Dinonaktifkan IPv6 untuk Stage koneksi.

Ukuran SDK Siaran: Android

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64-v8a	5,693 MB	13.838 MB
armeabi-v7a	4,964 MB	9,653 MB
x86_64	5.810 MB	14.422 MB
x86	6,067 MB	14.988 MB

Juli 23, 2025

Penegakan Metrik dan Batasan Real-Time Baru: Penerbit dan Langganan Serentak

Pada [tanggal 23 Juni](#), kami memperkenalkan dua kuota layanan baru yang dapat disesuaikan, untuk jumlah maksimum penerbit bersamaan dan langganan bersamaan di semua tahapan di Wilayah AWS. Hari ini kita mulai menegakkan kuota baru ini.

Juli 15, 2025

Batas Waktu Nyata Baru: Replikasi Peserta Serentak

Kami telah memperkenalkan kuota layanan baru yang tidak dapat disesuaikan, untuk jumlah maksimum replikasi bersamaan per peserta di semua tahapan di Wilayah AWS. Hal ini didokumentasikan dalam Service Quotas > [Other](#) Quotas.

Juli 10, 2025

SDK Siaran Amazon IVS: Android 1.32.1, iOS 1.32.1 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Android 1.32.1	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.32.1/android/</p> <ul style="list-style-type: none">• Dihapus <code>StageAudioConfiguration.enableEchoCancellation()</code>. Sebagai gantinya, gunakan <code>StageAudioManager</code> untuk mengaktifkan atau menonaktifkan pembatalan gema.• Memodifikasi <code>STUDIO</code> dan <code>SUBSCRIBE_ONLY</code> preset <code>StageAudioManager</code> untuk mematikan pembatalan gema. Jika

Platform	Unduhan dan Perubahan
	<p>Anda ingin menggunakan STUDIO pembatalan game, pertama-tama atur preset, lalu aktifkan pembatalan game untuk mengganti STUDIO preferensi default tanpa pembatalan game.</p> <ul style="list-style-type: none"> • Menambahkan <code>MixedDevice</code> API suite untuk mengomposisikan beberapa sumber gambar dan audio ke dalam satu <code>outputDevice</code>, yang dapat digunakan untuk mempublikasikan audio dan visual yang lebih kompleks ke panggung.
<p>SDK Siaran iOS 1.32.1</p>	<p>Unduh untuk streaming waktu nyata: https://broadcast.live-video.net/1.32.1/AmazonIVSBroadcast-Stages.xcFramework.zip</p> <p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.32.1/ios/</p> <ul style="list-style-type: none"> • Menambahkan <code>IVSMixedDevice</code> API suite untuk mengomposisikan beberapa sumber gambar dan audio ke dalam satu <code>outputIVSDevice</code>, yang dapat digunakan untuk mempublikasikan audio dan visual yang lebih kompleks ke panggung.

Ukuran SDK Siaran: Android

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64-v8a	5,692 MB	13.840 MB
armeabi-v7a	4,965 MB	9,655 MB
x86_64	5.810 MB	14.424 MB

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
x86	6,068 MB	14.990 MB

Ukuran SDK Siaran: iOS

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64	3,508 MB	7.900 MB

Juli 7, 2025

SDK Siaran IVS: Web 1.26.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Web 1.26.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-referensi</p> <ul style="list-style-type: none"> • Ditambahkan <code>requestQualityStats</code> ke <code>LocalStageStream</code>, yang memperlihatkan objek statistik video dan audio yang disederhanakan yang bersumber dari permintaan. <code>RTCStats</code> • Memperbaiki kebocoran websocket yang dapat terjadi selama penyiapan, menyebabkan kegagalan gabungan berikutnya. • Memperbaiki masalah di mana kesalahan 1302 salah muncul saat mencoba kembali operasi berlangganan atau publikasi yang gagal. • Peningkatan stabilitas coba lagi untuk berlangganan dan publikasikan koneksi saat

Platform	Unduhan dan Perubahan
	koneksi bergabung dalam status ERRORED atau CONNECTING.

Juni 23, 2025

Metrik dan Batas Waktu Nyata Baru: Penerbit dan Langganan Bersamaan

Kami telah memperkenalkan dua kuota layanan baru yang dapat disesuaikan, untuk jumlah maksimum penayang bersamaan dan langganan bersamaan di semua tahapan di Wilayah AWS. Mereka didokumentasikan dalam Service Quotas > [Other](#) Quotas. Kuota ini memberi Anda kontrol lebih besar atas total penggunaan di seluruh akun Anda. Sebelumnya, IVS memberlakukan batasan hanya pada jumlah penerbit dan pelanggan per tahap. Hal ini membuat sulit untuk menetapkan perlindungan di tingkat akun dan dapat menghasilkan penggunaan yang lebih tinggi dan biaya terkait dari yang diharapkan, terutama bagi pelanggan yang menciptakan banyak tahapan.

Catatan: Kami akan mulai menerapkan kuota baru ini pada tanggal 23 Juli, untuk memungkinkan 30 hari bagi Anda untuk meninjau penggunaan Anda dan meminta peningkatan kuota layanan jika diperlukan.

Kami juga menambahkan dua CloudWatch metrik baru, `ConcurrentPublishers` dan `ConcurrentSubscriptions`. Metrik ini membantu Anda memantau penggunaan di semua tahapan dan menilai apakah Anda mendekati batas default. Mereka didokumentasikan dalam Monitoring Real-Time Streaming > [CloudWatch Metrik](#). Sebaiknya atur [CloudWatch alarm](#) untuk mengingatkan Anda saat penggunaan Anda mendekati batas kuota.

Juni 20, 2025

E-RTMP Multitrack Video Insert Support

Anda dapat menggunakan video multitrack E-RTMP (Enhanced Real-Time Messaging Protocol) untuk mengirim beberapa kualitas video ke tahap IVS Anda. Fitur ini memungkinkan streaming bitrate adaptif, memungkinkan pemirsa menonton dengan kualitas terbaik untuk koneksi jaringan mereka. Lihat [E-RTMP Multitrack Video](#) dalam dokumentasi IVS RTMP Publishing.

Juni 16, 2025

SDK Siaran IVS: Web 1.25.1 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Web 1.25.1	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-referensi</p> <ul style="list-style-type: none">• Menghapus penegakan mesin NPM yang tidak disengaja dari v22. Semua versi node LTS didukung saat paket ditranspilasikan.

Juni 12, 2025

SDK Siaran Amazon IVS: Android 1.31.0, iOS 1.31.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Android 1.31.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.31.0/android/</p> <ul style="list-style-type: none">• Perbaiki bug dan peningkatan stabilitas.
SDK Siaran iOS 1.31.0	<p>Unduh untuk streaming waktu nyata: https://broadcast.live-video.net/1.31.0/AmazonIVSBroadcast -Stages.xcFramework.zip</p> <p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.31.0/ios/</p> <ul style="list-style-type: none">• Perbaiki bug dan peningkatan stabilitas.

Ukuran SDK Siaran: Android

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64-v8a	5,579 MB	13,594 MB
armeabi-v7a	4,864 MB	9,473 MB
x86_64	5,697 MB	14.173 MB
x86	5,951 MB	14.724 MB

Ukuran SDK Siaran: iOS

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64	3,431 MB	7,732 MB

Juni 12, 2025

SDK Siaran IVS: Web 1.25.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
Siaran Web SDK 1.25.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-referensi</p> <ul style="list-style-type: none"> • Memperbaiki bug di mana pesan SEI mungkin gagal dikirim setelah peserta jarak jauh mengalami ERROR status. • Memperbaiki bug di mana beberapa aliran tahap jarak jauh mungkin dikembalikan saat acara <code>STAGE_STREAM_MUTE_CHANGED</code> panggung dipanggil. • Memperbaiki bug yang tidak <code>STAGE_PARTICIPANT_STREAMS_REMOVED</code>

Platform	Unduhan dan Perubahan
	dipanggil untuk aliran yang telah mengalami kesalahan.

29 Mei 2025

Replikasi Peserta

Replikasi peserta memungkinkan Anda untuk menyalin peserta dari satu tahap ke tahap lainnya. Ini berguna ketika Anda ingin peserta yang sama muncul dalam beberapa tahap pada saat yang sama, memungkinkan interaksi lintas tahap. Untuk perubahan dokumentasi, lihat [Riwayat Dokumen](#) (Panduan Pengguna dan tabel Referensi API).

26 Mei 2025

SDK Siaran Amazon IVS: Android 1.30.1 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Android 1.30.1	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.30.1/android/</p> <ul style="list-style-type: none"> Memperbaiki low-microphone-volume bug pada beberapa perangkat Android saat menggunakan mikrofon yang dikelola SDK <code>DeviceDiscovery</code> dengan preset audio. STUDIO

Ukuran SDK Siaran: Android

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64-v8a	5,579 MB	13,592 MB
armeabi-v7a	4,863 MB	9,472 MB

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
x86_64	5,696 MB	14.171 MB
x86	5,950 MB	14.722 MB

15 Mei 2025

SDK Siaran IVS: Web 1.24.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Web 1.24.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-referensi</p> <ul style="list-style-type: none"> • Memori tetap bocor saat meninggalkan dan bergabung kembali dengan panggung.

15 Mei 2025

SDK Siaran Amazon IVS: Android 1.30.0, iOS 1.30.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Android 1.30.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.30.0/android/</p> <ul style="list-style-type: none"> • Perbaikan bug dan peningkatan stabilitas.
SDK Siaran iOS 1.30.0	<p>Unduh untuk streaming waktu nyata: https://broadcast.live-video.net/1.30.0/AmazonIVSBroadcast -Stages.xcFramework.zip</p> <p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.30.0/ios/</p>

Platform	Unduhan dan Perubahan
	<ul style="list-style-type: none"> Perbaiki bug dan peningkatan stabilitas.

Ukuran SDK Siaran: Android

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64-v8a	5,571 MB	13,577 MB
armeabi-v7a	4,857 MB	9,462 MB
x86_64	5,691 MB	14.156 MB
x86	5,944 MB	14.708 MB

Ukuran SDK Siaran: iOS

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64	3,430 MB	7,732 MB

2 Mei 2025

SDK Siaran IVS: Web 1.23.1 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Web 1.23.1	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-referensi</p> <ul style="list-style-type: none"> Memperbaiki masalah saat peserta bergabung dengan acara selalu terjadi sebelum <code>join()</code> diselesaikan. Memperbaiki masalah di mana peserta lokal dilaporkan secara keliru sebagai peserta

Platform	Unduhan dan Perubahan
	jarak jauh ketika pergi dan bergabung kembali secara berurutan.

17 April 2025

SDK Siaran Amazon IVS: Android 1.29.0, iOS 1.29.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Android 1.29.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.29.0/android/</p> <ul style="list-style-type: none"> • Menambahkan fitur kontrol penerbit simulcast . Lihat “Mengkonfigurasi Layered Encoding (Publisher)” di Panduan SDK Broadcast Android. • Perbaiki bug dan peningkatan stabilitas.
SDK Siaran iOS 1.29.0	<p>Unduh untuk streaming waktu nyata: https://broadcast.live-video.net/1.29.0/AmazonIVSBroadcast -Stages.xcFramework.zip</p> <p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.29.0/ios/</p> <ul style="list-style-type: none"> • Menambahkan fitur kontrol penerbit simulcast . Lihat “Mengkonfigurasi Layered Encoding (Publisher)” di Panduan SDK Siaran iOS. • Perbaiki bug dan peningkatan stabilitas.

Ukuran SDK Siaran: Android

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64-v8a	5,566 MB	13,546 MB
armeabi-v7a	4,853 MB	9,444 MB
x86_64	5,681 MB	14.119 MB
x86	5,939 MB	14.674 MB

Ukuran SDK Siaran: iOS

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64	3,429 MB	7,715 MB

17 April 2025

SDK Siaran IVS: Web 1.23.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
Siaran Web SDK 1.23.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-referensi</p> <ul style="list-style-type: none"> Menambahkan fitur kontrol penerbit simulcast . Lihat “Mengkonfigurasi Layered Encoding (Publisher)” di Panduan SDK Siaran Web. Peningkatan waktu untuk mempublikasikan latensi. Ini berdampak pada waktu PUBLISHED acara. Memperbaiki bug di mana SDK memicu kesalahan kategori gabungan melalui panggilan balik ERROR saat koneksi

Platform	Unduhan dan Perubahan
	<p>ke panggung terputus tetapi berpotensi dapat dipulihkan (khususnya, FAILED dan TIMEOUT kesalahan untuk kategori). JOIN_ERROR</p> <ul style="list-style-type: none"> • Memperbaiki bug dengan <code>insertSeiMessage</code> operasi di mana penyegaran strategi dapat mengakibatkan pemanggilan berikutnya <code>insertSeiMessage</code> gagal mengirim pesan SEI.

April 2, 2025

Kuota Baru: Komposisi Per Tahap

Kami menambahkan kuota baru, untuk komposisi bersamaan maksimum yang diizinkan per tahap. Ini didokumentasikan dalam Service Quotas > [Other](#) Quotas.

Maret 20, 2025

SDK Siaran Amazon IVS: Android 1.28.1, iOS 1.28.1 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Android 1.28.1	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.28.1/android/</p> <ul style="list-style-type: none"> • Perbaiki bug dan peningkatan stabilitas.
SDK Siaran iOS 1.28.1	<p>Unduh untuk streaming waktu nyata: https://broadcast.live-video.net/1.28.1/AmazonIVSBroadcast -Stages.xcFramework.zip</p> <p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.28.1/ios/</p>

Platform	Unduhan dan Perubahan
	<ul style="list-style-type: none"> Perbaikan bug dan peningkatan stabilitas.

Ukuran SDK Siaran: Android

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64-v8a	5,613 MB	13.760 MB
armeabi-v7a	4,885 MB	9,558 MB
x86_64	5,728 MB	14.342 MB
x86	5,987 MB	14.923 MB

Ukuran SDK Siaran: iOS

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64	3,417 MB	7,698 MB

Maret 20, 2025

SDK Siaran IVS: Web 1.22.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
Siaran Web SDK 1.22.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-referensi</p> <ul style="list-style-type: none"> Ditambahkan null sebagai tipe pengembalian yang valid ke metode strategi preferred LayerForStream. Memperbaiki bug yang tidak preferred LayerForStream dipanggil lagi jika

Platform	Unduhan dan Perubahan
	<p>lapisan baru tersedia setelah streaming dimulai.</p> <ul style="list-style-type: none"> • Memperbaiki bug <code>stream.getHighestQualityLayer</code> yang tidak memilih lapisan kualitas tertinggi setelah streaming dimulai.

Maret 19, 2025

SDK Siaran Amazon IVS: Android 1.27.2, iOS 1.27.2 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
<p>SDK Siaran Android 1.27.2</p>	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.27.2/android/</p> <ul style="list-style-type: none"> • Memperbaiki regresi kebocoran sumber daya yang memengaruhi beberapa perangkat saat membuat 50 tahapan atau lebih. • Memperbaiki regresi yang dapat menyebabkan peningkatan laju pembekuan video saat menggunakan perangkat lunak penerbitan pihak ketiga.
<p>SDK Siaran iOS 1.27.2</p>	<p>Unduh untuk streaming waktu nyata: https://broadcast.live-video.net/1.27.2/AmazonIVSBroadcast-Stages.xcFramework.zip</p> <p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.27.2/ios/</p> <ul style="list-style-type: none"> • Memperbaiki regresi yang dapat menyebabkan peningkatan laju pembekuan video saat

Platform	Unduhan dan Perubahan
	menggunakan perangkat lunak penerbitan pihak ketiga.

Ukuran SDK Siaran: Android

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64-v8a	5.700 MB	14.197 MB
armeabi-v7a	4,945 MB	9,879 MB
x86_64	5.810 MB	14.802 MB
x86	6,073 MB	15.412 MB

Ukuran SDK Siaran: iOS

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64	3,622 MB	8,584 MB

Maret 13, 2025

Durasi Segmen Target

Rilis ini menambah API streaming real-time IVS, untuk memungkinkan Anda menentukan durasi target untuk segmen rekaman yang dihasilkan saat menggunakan rekaman komposit atau merekam peserta panggung. Untuk perubahan API tertentu, lihat [Riwayat Dokumen](#) (tabel Panduan Pengguna dan Referensi API).

Maret 6, 2025

Jahitan Rekaman Peserta Individu

Ini adalah rilis pertama dari fungsionalitas baru. Jika tahap Anda dikonfigurasi untuk perekaman peserta individu, Anda sekarang dapat menentukan jendela waktu di mana, jika penerbit panggung terputus dari panggung dan kemudian menyambung kembali, IVS mencoba merekam ke awalan S3 yang sama dengan sesi sebelumnya. Dengan kata lain, jika penerbit terputus dan kemudian menyambung kembali dalam interval yang ditentukan, beberapa rekaman dianggap sebagai rekaman tunggal dan digabungkan. Untuk perubahan dokumentasi, lihat [Riwayat Dokumen](#) (baik tabel Panduan Pengguna dan Referensi API).

Maret 3, 2025

SDK Siaran Amazon IVS: iOS 1.27.1 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran iOS 1.27.1	<p>Unduh untuk streaming waktu nyata: https://broadcast.live-video.net/1.27.1/AmazonIVSBroadcast -Stages.xcFramework.zip</p> <p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.27.1/ios/</p> <ul style="list-style-type: none">• Peningkatan kinerja fokus untuk objek yang dipegang dekat dengan kamera saat menggunakan lensa ultra lebar pada perangkat Pro.

Ukuran SDK Siaran: iOS

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64	3,625 MB	8,601 MB

Februari 20, 2025

SDK Siaran Amazon IVS: Android 1.27.0, iOS 1.27.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Android 1.27.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.27.0/android/</p> <ul style="list-style-type: none"> Perbaikan bug dan peningkatan stabilitas.
SDK Siaran iOS 1.27.0	<p>Unduh untuk streaming waktu nyata: https://broadcast.live-video.net/1.27.0/AmazonIVSBroadcast -Stages.xcFramework.zip</p> <p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.27.0/ios/</p> <ul style="list-style-type: none"> Perbaikan bug dan peningkatan stabilitas.

Ukuran SDK Siaran: Android

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64-v8a	5.700 MB	14.197 MB
armeabi-v7a	4,944 MB	9,879 MB
x86_64	5,809 MB	14.802 MB
x86	6,073 MB	15.412 MB

Ukuran SDK Siaran: iOS

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64	3,625 MB	8,601 MB

Februari 20, 2025

SDK Siaran IVS: Web 1.21.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
Siaran Web SDK 1.21.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-referensi</p> <ul style="list-style-type: none">Jenis <code>preferredLayerForStream</code> strategi yang diperbarui untuk disertakan <code>null</code>, yang merupakan pengembalian yang valid.Memperbaiki kesalahan TypeScript kompilasi saat <code>Tsconfig skipLibCheck</code> disetel ke <code>false</code>. <p>Catatan: Sebagai bagian dari rilis ini, tipe telah dikonsolidasikan menjadi satu rollup. Jika aplikasi mengimpor tipe bersarang berdasarkan jalur, kesalahan dapat terjadi. Jika kesalahan memang terjadi, ubah impor menjadi sederhana <code>'amazon-ivs-broadcast'</code>.</p>

Januari 30, 2025

SDK Siaran Amazon IVS: Android 1.26.0, iOS 1.26.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Android 1.26.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.26.0/android/</p> <ul style="list-style-type: none"> Perbaikan bug dan peningkatan stabilitas.
SDK Siaran iOS 1.26.0	<p>Unduh untuk streaming waktu nyata: https://broadcast.live-video.net/1.26.0/AmazonIVSBroadcast -Stages.xcFramework.zip</p> <p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.26.0/ios/</p> <ul style="list-style-type: none"> Perbaikan bug dan peningkatan stabilitas.

Ukuran SDK Siaran: Android

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64-v8a	5,695 MB	14.186 MB
armeabi-v7a	4,939 MB	9,872 MB
x86_64	5.804 MB	14.790 MB
x86	6,065 MB	15.398 MB

Ukuran SDK Siaran: iOS

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64	3,624 MB	8,601 MB

Januari 23, 2025

SDK Siaran IVS: Web 1.20.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Web 1.20.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-referensi</p> <ul style="list-style-type: none"> Menambahkan <code>insertSeiMessage</code> metode <code>LocalStageStream</code> untuk mengaktifkan penyisipan payload Informasi Peningkatan Tambahan (SEI) ke dalam aliran video penerbitan. Lihat Informasi Tambahan yang Ditingkatkan di SDK Siaran IVS: Panduan Web.

Desember 12, 2024

SDK Siaran Amazon IVS: Android 1.25.0, iOS 1.25.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Android 1.25.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.25.0/android/</p> <ul style="list-style-type: none"> Menambahkan fitur kontrol simulcast. Lihat Mengkonfigurasi Layered Encoding dengan

Platform	Unduhan dan Perubahan
	<p data-bbox="862 212 1438 289">Simulcast (Subscriber) di Pengoptimalan Streaming.</p> <ul data-bbox="829 317 1503 846" style="list-style-type: none"><li data-bbox="829 317 1503 590">• Made SEI (Supplemental Enhanced Information) payload tersedia untuk pelanggan dengan bidang baru pada objek. ImageDeviceFrame Lihat Mendapatkan Informasi Peningkatan Tambahan (SEI) di SDK Siaran IVS: Panduan Android.<li data-bbox="829 611 1503 789">• Menambahkan <code>SubscribeConfiguration::setInitialGain</code> metode untuk memungkinkan konfigurasi nilai gain awal untuk aliran audio yang masuk.<li data-bbox="829 810 1503 846">• Perbaiki bug dan peningkatan stabilitas.

Platform	Unduhan dan Perubahan
<p>SDK Siaran iOS 1.25.0</p>	<p>Unduh untuk streaming waktu nyata: https://broadcast.live-video.net/1.25.0/AmazonIVSBroadcast -Stages.xcFramework.zip</p> <p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.25.0/ios/</p> <ul style="list-style-type: none"> • Menambahkan fitur kontrol simulcast. Lihat Mengkonfigurasi Layered Encoding dengan Simulcast (Subscriber) di Pengoptimalan Streaming. • Made SEI (Supplemental Enhanced Information) payload tersedia untuk pelanggan dengan bidang baru pada objek. IVSImage DeviceFrame Lihat Mendapatkan Informasi Peningkatan Tambahan (SEI) di SDK Siaran IVS: Panduan iOS. • Menambahkan <code>IVSSubscribeConfiguration.initialGain</code> metode untuk memungkinkan konfigurasi nilai gain awal untuk aliran audio yang masuk. • Perbaiki bug dan peningkatan stabilitas.

Ukuran SDK Siaran: Android

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64-v8a	5,677 MB	14.103 MB
armeabi-v7a	4,905 MB	9,791 MB
x86_64	5,786 MB	14.725 MB
x86	6,030 MB	15.302 MB

Ukuran SDK Siaran: iOS

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64	3,625 MB	8,585 MB

Desember 12, 2024

SDK Siaran IVS: Web 1.19.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Web 1.19.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-referensi</p> <ul style="list-style-type: none"> Menambahkan fitur kontrol simulcast. Lihat Mengkonfigurasi Layered Encoding dengan Simulcast (Subscriber) di Pengoptimalan Streaming. Perbaikan bug dan peningkatan stabilitas.

Desember 10, 2024

Konfigurasi Thumbnail Streaming Waktu Nyata

Rilis ini memungkinkan enable/disable Anda merekam thumbnail untuk sesi langsung dan memodifikasi interval di mana thumbnail dihasilkan untuk sesi langsung. Ini adalah rilis pertama dari fungsi baru ini. Lihat:

- [Rekaman Peserta Individu](#) — Kami memperbarui contoh dan informasi metadata JSON, dan kami menambahkan informasi harga dan “Rekaman Khusus Thumbnail.”
- [Rekaman Komposit](#) - Kami memperbarui contoh dan informasi metadata JSON, dan kami menambahkan informasi harga.
- [Referensi API RT](#) - Kami membuat beberapa perubahan:

- Memodifikasi DestinationConfiguration objek S3: ditambahkan thumbnailConfigurations. Hal ini mempengaruhi GetComposition respon dan StartComposition permintaan dan respon.
- Memodifikasi AutoParticipantRecordingConfiguration objek: ditambahkan thumbnailConfiguration dan ditambahkan NONE sebagai nilai yang valid untuk mediaTypes. Hal ini mempengaruhi CreateStage permintaan dan respon, GetStage respon, dan UpdateStage permintaan dan respon.
- Menambahkan dua objek: CompositionThumbnailConfiguration dan ParticipantThumbnailConfiguration.

November 13, 2024

SDK Siaran Amazon IVS: Android 1.24.0, iOS 1.24.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Android 1.24.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.24.0/android/</p> <ul style="list-style-type: none"> • Perbaiki bug dan peningkatan stabilitas.
SDK Siaran iOS 1.24.0	<p>Unduh untuk streaming waktu nyata: https://broadcast.live-video.net/1.24.0/AmazonIVSBroadcast-Stages.xcFramework.zip</p> <p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.24.0/ios/</p> <ul style="list-style-type: none"> • Perbaiki bug dan peningkatan stabilitas.

Ukuran SDK Siaran: Android

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64-v8a	5,521 MB	13.791 MB

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
armeabi-v7a	4,789 MB	9,623 MB
x86_64	5,718 MB	14.709 MB
x86	5,933 MB	15.163 MB

Ukuran SDK Siaran: iOS

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64	3,589 MB	8,466 MB

November 12, 2024

SDK Siaran IVS: Web 1.18.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Web 1.18.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-referensi</p> <ul style="list-style-type: none"> • Menambahkan acara baru untuk membuat muatan SEI (Supplemental Enhanced Information) tersedia untuk pelanggan. • Memperbaiki pengecualian yang akan terjadi selama permintaan unpublish dan berhenti berlangganan. • Memperbaiki kondisi balapan di mana bergabung dan pergi dengan cepat akan menyebabkan kesalahan bagi peserta lain.

Oktober 10, 2024

SDK Siaran IVS: Web 1.17.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Web 1.17.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-referensi</p> <ul style="list-style-type: none">• Perbaiki bug minor.

Oktober 10, 2024

SDK Siaran Amazon IVS: Android 1.23.0, iOS 1.23.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Android 1.23.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.23.0/android/</p> <ul style="list-style-type: none">• Dengan rilis ini kami juga mulai menerbitkan versi SDK siaran Android yang mencakup simbol debug. Lihat Menggunakan SDK dengan Simbol Debug.• Perbaiki bug minor.
SDK Siaran iOS 1.23.0	<p>Unduh untuk streaming waktu nyata: https://broadcast.live-video.net/1.23.0/AmazonIVSBroadcast -Stages.xcFramework.zip</p> <p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.23.0/ios/</p> <ul style="list-style-type: none">• Perbaiki bug minor.

Ukuran SDK Siaran: Android

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64-v8a	5,432 MB	13,560 MB
armeabi-v7a	4,707 MB	9,451 MB
x86_64	5,626 MB	14.459 MB
x86	5,838 MB	14.908 MB

Ukuran SDK Siaran: iOS

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64	3,542 MB	8,316 MB

September 11, 2024

SDK Siaran Amazon IVS: Android 1.22.0, iOS 1.22.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Android 1.22.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.22.0/android/</p> <ul style="list-style-type: none">• Memperbaiki bug di mana perangkat Android tertentu menampilkan bingkai hitam di pratinjau setelah beralih input kamera.• Perbaiki bug minor.
SDK Siaran iOS 1.22.0	<p>Unduh untuk streaming waktu nyata: https://broadcast.live-video.net/1.22.0/AmazonIVSBroadcast -Stages.xcFramework.zip</p>

Platform	Unduhan dan Perubahan
	Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.22.0/ios/ <ul style="list-style-type: none"> Perbaikan bug minor.

Ukuran SDK Siaran: Android

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64-v8a	5,359 MB	13.392 MB
armeabi-v7a	4,636 MB	9,325 MB
x86_64	5,548 MB	14.268 MB
x86	5,754 MB	14.710 MB

Ukuran SDK Siaran: iOS

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64	3,488 MB	8,199 MB

September 11, 2024

SDK Siaran IVS: Web 1.16.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Web 1.16.0	Dokumentasi referensi: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-referensi <ul style="list-style-type: none"> Perbaikan bug minor.

September 9, 2024

Tertelan RTMP

Sebagai alternatif untuk menggunakan SDK siaran IVS, Anda sekarang dapat mempublikasikan video ke tahap IVS dari sumber RTMP (selain WHIP, yang sudah didukung). Untuk perubahan dokumentasi, lihat [Riwayat Dokumen](#) (baik tabel Panduan Pengguna dan Referensi API).

Agustus 19, 2024

Terbitkan/Berlangganan Dalam Konsol

Anda sekarang dapat menerbitkan dan berlangganan dari konsol IVS. Dalam Memulai Streaming Waktu Nyata IVS, lihat [Mempublikasikan dan Berlangganan Video](#).

Agustus 15, 2024

SDK Siaran IVS: Web 1.15.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Web 1.15.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-referensi</p> <ul style="list-style-type: none">• Memperbaiki kondisi balapan yang memengaruhi kualitas media penerbit ketika <code>join()</code> dipanggil berulang kali. Memanggil <code>join()</code> berturut-turut tidak lagi memicu ulang <code>STAGE_PARTICIPANT_JOINED</code> acara, bersama dengan perubahan status publikasi dan streaming yang menyertainya.• Memperbaiki bug yang menyebabkan masalah mengurai token peserta saat karakter non-teks digunakan di bidang <code>tokenattributes</code> .

Platform	Unduhan dan Perubahan
	<ul style="list-style-type: none"> Menambahkan metode untuk mengkonfigurasi pelanggan peserta. Awalnya, Anda hanya dapat mengonfigurasi penundaan minimum jitter-buffer. Lihat dokumentasi referensi SDK, Konfigurasi untuk Berlangganan Peserta dalam Panduan SDK Siaran Web, dan Mengubah Buffer Jitter Pelanggan di Pengoptimalan Streaming. MinDelay

Agustus 15, 2024

SDK Siaran Amazon IVS: Android 1.21.0, iOS 1.21.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Android 1.21.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.21.0/android/</p> <ul style="list-style-type: none"> Memperbaiki bug yang memengaruhi perangkat dengan MT6765 chipset, di mana pratinjau pelanggan membuat bingkai hitam dalam beberapa keadaan. Menambahkan metode untuk mengkonfigurasi pelanggan peserta. Awalnya, Anda hanya dapat mengonfigurasi penundaan minimum jitter-buffer. Lihat dokumentasi referensi SDK, Konfigurasi untuk Berlangganan Peserta di Panduan SDK Broadcast Android, dan Mengubah Buffer Jitter Pelanggan di Pengoptimalan Streaming. MinDelay Perbaiki bug minor.

Platform	Unduhan dan Perubahan
SDK Siaran iOS 1.21.0	<p>Unduh untuk streaming waktu nyata: https://broadcast.live-video.net/1.21.0/AmazonIVSBroadcast -Stages.xcFramework.zip</p> <p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.21.0/ios/</p> <ul style="list-style-type: none"> Menambahkan metode untuk mengkonfigurasi pelanggan peserta. Awalnya, Anda hanya dapat mengonfigurasi penundaan minimum jitter-buffer. Lihat dokumentasi referensi SDK, Konfigurasi untuk Berlangganan Peserta di Panduan SDK Siaran iOS, dan Mengubah Buffer Jitter Pelanggan di Pengoptimalan Streaming. MinDelay Perbaikan bug minor.

Ukuran SDK Siaran: Android

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64-v8a	5.350 MB	13.378 MB
armeabi-v7a	4,628 MB	9,312 MB
x86_64	5,538 MB	14.253 MB
x86	5,744 MB	14.694 MB

Ukuran SDK Siaran: iOS

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64	3,485 MB	8,199 MB

Juli 18, 2024

SDK Siaran IVS: Web 1.14.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Web 1.14.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-referensi</p> <ul style="list-style-type: none">• Peningkatan dokumentasi API.• Outlier statistik video dan audio tetap dilaporkan selama penyetelan ulang koneksi.• Pembaruan ketergantungan kecil.

Juli 18, 2024

SDK Siaran Amazon IVS: Android 1.20.0, iOS 1.20.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Android 1.20.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.20.0/android</p> <ul style="list-style-type: none">• Memperbaiki bug yang mencegah Broadcast SDK berjalan di Chromebook dengan prosesor Intel.• Perbaiki bug minor.
SDK Siaran iOS 1.20.0	<p>Unduh untuk streaming waktu nyata: https://broadcast.live-video.net/1.20.0/AmazonIVSBroadcast -Stages.xcFramework.zip</p> <p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.20.0/ios</p>

Platform	Unduhan dan Perubahan
	<ul style="list-style-type: none">Perbaiki bug minor.

Ukuran SDK Siaran: Android

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64-v8a	5.318 MB	13,299 MB
armeabi-v7a	4,605 MB	9,254 MB
x86_64	5,507 MB	14.168 MB
x86	5,715 MB	14.608 MB

Ukuran SDK Siaran: iOS

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64	3,465 MB	8,164 MB

Juni 26, 2024

Hasilkan Token Peserta dengan Pasangan Kunci

Anda sekarang dapat menghasilkan token peserta pada aplikasi server Anda sendiri dengan menggunakan key pair. Ini memungkinkan Anda untuk menghindari menelepon `CreateParticipantToken` setiap kali Anda membutuhkan token peserta. Untuk perubahan dokumentasi, lihat [Riwayat Dokumen](#) (baik tabel Panduan Pengguna dan Referensi API).

Juni 20, 2024

Rekaman Peserta Individu

Rekaman peserta individu memungkinkan pelanggan streaming real-time IVS merekam penerbit panggung IVS satu per satu ke dalam ember S3. Lihat [Perekaman](#), [Perekaman Peserta Individu](#), dan perubahan dalam [Referensi API Streaming Waktu Nyata](#). (Untuk perubahan dokumentasi tertentu, lihat [Riwayat Dokumen](#).)

Juni 13, 2024

SDK Siaran Amazon IVS: Android 1.19.0, iOS 1.19.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Android 1.19.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.19.0/android</p> <ul style="list-style-type: none">Versi Android terbaru memerlukan ikon dalam notifikasi yang ditampilkan saat menangkap layar. Jika diinginkan, Anda sekarang dapat menyesuaikan ikon dengan memanggil <code>setSmallIcon</code> yang <code>Notification.Builder</code> dikembalikan oleh <code>Session#createServiceNotificationBuilder</code>.Peningkatan waktu pemulihan koneksi pada perangkat yang beralih dari wifi ke koneksi seluler. Perubahan ini membutuhkan <code>CHANGE_NETWORK_STATE</code> izin.
SDK Siaran iOS 1.19.0	<p>Unduh untuk streaming waktu nyata: https://broadcast.live-video.net/1.19.0/AmazonIVSBroadcast-Stages.xcFramework.zip</p>

Platform	Unduhan dan Perubahan
	Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.19.0/ios <ul style="list-style-type: none"> Perbaiki bug minor.

Ukuran SDK Siaran: Android

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64-v8a	5.304 MB	13.340 MB
armeabi-v7a	4,598 MB	9,299 MB
x86_64	5,495 MB	14.207 MB
x86	5,694 MB	14.625 MB

Ukuran SDK Siaran: iOS

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64	3,393 MB	7,949 MB

Juni 13, 2024

SDK Siaran IVS: Web 1.13.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Web 1.13.0	Dokumentasi referensi: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-referensi <ul style="list-style-type: none"> Memperbarui durasi perilaku perubahan acara untuk <code>StageEvents.STAGE_</code>

Platform	Unduhan dan Perubahan
	<p>PARTICIPANT_SUBSCRIBE_STATE_CHANGED dan StageEvents.STAGE_PARTICIPANT_PUBLISH_STATE_CHANGED . Peserta sekarang tetap berada di ATTEMPTING_PUBLISH negara bagian ATTEMPTING_SUBSCRIBE atau untuk waktu yang lebih lama, sampai ERRORED acara dipecat.</p> <ul style="list-style-type: none"> • Menambahkan StageEvents.ERROR acara untuk mendengarkan kesalahan yang dihadapi oleh SDK. Lihat Penanganan Kesalahan di SDK Siaran Real-Time: Panduan Web untuk informasi selengkapnya.

Mei 20, 2024

SDK Siaran IVS: Web 1.12.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
<p>Siaran Web SDK 1.12.0</p>	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-referensi</p> <ul style="list-style-type: none"> • Peningkatan penanganan coba lagi untuk operasi publikasi dan berlangganan. • Analisis yang ditingkatkan, khususnya latensi dan pengukuran kualitas audio.

16 Mei 2024

SDK Siaran Amazon IVS: Android 1.18.0, iOS 1.18.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Android 1.18.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.18.0/android</p> <ul style="list-style-type: none"> • SDK sekarang mengirimkan kode kesalahan tertentu saat Tahap yang terhubung dihapus oleh bidang kontrol AWS, atau saat token yang digunakan dicabut. • Perbaiki bug minor.
SDK Siaran iOS 1.18.0	<p>Unduh untuk streaming waktu nyata: https://broadcast.live-video.net/1.18.0/AmazonIVSBroadcast-Stages.xcFramework.zip</p> <p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.18.0/ios</p> <ul style="list-style-type: none"> • SDK sekarang mengirimkan kode kesalahan tertentu saat Tahap yang terhubung dihapus oleh bidang kontrol AWS, atau saat token yang digunakan dicabut. • Menambahkan <code>IVSCamera setVideoZoomFactor</code> metode dan <code>IVSCamera Delegate</code> metode terkait.

Ukuran SDK Siaran: Android

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64-v8a	5.275 MB	13.279 MB

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
armeabi-v7a	4,573 MB	9,254 MB
x86_64	5,472 MB	14.142 MB
x86	5,664 MB	14.554 MB

Ukuran SDK Siaran: iOS

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64	3,393 MB	7,916 MB

6 Mei 2024

SDK Siaran IVS: Web 1.11.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Web 1.11.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-referensi</p> <ul style="list-style-type: none"> • Memperbaiki kasus tepi di mana SDK tidak mencoba memulihkan di atas panggungDISCONNECT . • Memperbarui pesan kesalahan untuk kesalahan <code>join()</code> batas waktu. Alih-alih "InitialConnectTimedOut setelah 10 detik," SDK sekarang mengembalikan "Waktu operasi habis."

April 30, 2024

SDK Siaran IVS: Web 1.10.1 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Web 1.10.1	Dokumentasi referensi: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-referensi <ul style="list-style-type: none">• Perbaiki bug minor.

April 30, 2024

SDK Siaran Amazon IVS: Android 1.15.2, iOS 1.15.2 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Android 1.15.2	Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.15.2/android <ul style="list-style-type: none">• Perbaiki bug minor. Tingkatkan ke versi ini hanya jika Anda memiliki alasan khusus untuk melakukannya; jika tidak, gunakan versi tertinggi yang dirilis.
SDK Siaran iOS 1.15.2	Unduh untuk streaming waktu nyata: https://broadcast.live-video.net/1.15.2/AmazonIVSBroadcast -Stages.xcFramework.zip Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.15.2/ios <ul style="list-style-type: none">• Perbaiki bug minor. Tingkatkan ke versi ini hanya jika Anda memiliki alasan khusus

Platform	Unduhan dan Perubahan
	untuk melakukannya; jika tidak, gunakan versi tertinggi yang dirilis.

Ukuran SDK Siaran: Android

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64-v8a	5,244 MB	13.198 MB
armeabi-v7a	4,543 MB	9,192 MB
x86_64	5,437 MB	14.051 MB
x86	5,631 MB	14.461 MB

Ukuran SDK Siaran: iOS

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64	3,359 MB	7,836 MB

April 22, 2024

SDK Siaran Amazon IVS: Android 1.17.0, iOS 1.17.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Android 1.17.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.17.0/android</p> <ul style="list-style-type: none"> Memperbaiki crash langka yang dapat terjadi saat penerbitan.

Platform	Unduhan dan Perubahan
SDK Siaran iOS 1.17.0	<p>Unduh untuk streaming waktu nyata: https://broadcast.live-video.net/1.17.0/AmazonIVSBroadcast -Stages.xcFramework.zip</p> <p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.17.0/ios</p> <ul style="list-style-type: none"> • AmazonIVSBroadcast Kerangka kerja sekarang mencakup manifes privasi, seperti yang dipersyaratkan oleh Apple.

Ukuran SDK Siaran: Android

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64-v8a	5.273 MB	13.275 MB
armeabi-v7a	4,571 MB	9,251 MB
x86_64	5,468 MB	14.137 MB
x86	5,662 MB	14,549 MB

Ukuran SDK Siaran: iOS

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64	3,388 MB	7,916 MB

Maret 21, 2024

SDK Siaran Amazon IVS: Android 1.16.0, iOS 1.16.0, Web 1.10.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Web 1.10.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-referensi</p> <ul style="list-style-type: none">• Memperbaiki kesalahan intermiten saat membersihkan koneksi setelah berhenti berlangganan atau meninggalkan panggung.
SDK Siaran Android 1.16.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.16.0/android</p> <ul style="list-style-type: none">• Memperbaiki pembekuan pratinjau pada varian Exynos perangkat Samsung dengan Android 14.• Menambahkan fungsi untuk menanyakan kemampuan zoom kamera dan mengatur faktor zoom.
SDK Siaran iOS 1.16.0	<p>Unduh untuk streaming waktu nyata: https://broadcast.live-video.net/1.16.0/AmazonIVSBroadcast-Stages.xcFramework.zip</p> <p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.16.0/ios</p> <ul style="list-style-type: none">• Perbaiki bug minor.

Ukuran SDK Siaran: Android

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64-v8a	5,253 MB	13,21 MB
armeabi-v7a	4,551 MB	9.204 MB
x86_64	5,447 MB	14.070 MB
x86	5,640 MB	14.480 MB

Ukuran SDK Siaran: iOS

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64	3,361 MB	7,836 MB

Maret 13, 2024

SDK Siaran Amazon IVS: Android 1.15.1, iOS 1.15.1 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
SDK Siaran Android 1.15.1	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.15.1/android</p> <ul style="list-style-type: none"> • Memperbaiki crash yang jarang terjadi saat berlangganan peserta jarak jauh.
SDK Siaran iOS 1.15.1	<p>Unduh untuk streaming waktu nyata: https://broadcast.live-video.net/1.15.1/AmazonIVSBroadcast -Stages.xcFramework.zip</p>

Platform	Unduhan dan Perubahan
	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.15.1/ios</p> <ul style="list-style-type: none"> Memperbaiki crash yang jarang terjadi saat berlangganan peserta jarak jauh.

Ukuran SDK Siaran: Android

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64-v8a	5,243 MB	13.194 MB
armeabi-v7a	4,541 MB	9.188 MB
x86_64	5,628 MB	14.455 MB
x86	5,434 MB	14.046 MB

Ukuran SDK Siaran: iOS

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64	3,358 MB	7,820 MB

Maret 13, 2024

Pembaruan API Komposisi Sisi Server

Kami memperkenalkan properti baru ke `GridConfiguration` dan `picture-in-picture` tata letak baru, meningkatkan opsi penyesuaian untuk komposisi. Untuk perubahan dokumentasi tertentu, lihat [Riwayat Dokumen](#) (lihat tabel perubahan Referensi API).

Penting: Pastikan aplikasi Anda tidak bergantung pada fitur spesifik dari tata letak saat ini, seperti ukuran dan posisi ubin. Perbaikan visual pada tata letak dapat diperkenalkan kapan saja.

8 Maret 2024

Pembaruan Tata Letak Komposisi Sisi Server

Hari ini kami mengaktifkan perubahan pada tata letak kisi default yang dijelaskan dalam entri [7 Februari 2024](#).

Februari 22, 2024

SDK Siaran Amazon IVS: Android 1.15.0, iOS 1.15.0, Web 1.9.0 (Streaming Waktu Nyata)

Platform	Unduhan dan Perubahan
Siaran Web SDK 1.9.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-referensi</p> <ul style="list-style-type: none"> • Peningkatan penanganan kesalahan internal.
SDK Siaran Android 1.15.0	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.15.0/android</p> <ul style="list-style-type: none"> • Perbaiki bug minor.
SDK Siaran iOS 1.15.0	<p>Unduh untuk streaming waktu nyata: https://broadcast.live-video.net/1.15.0/AmazonIVSBroadcast-Stages.xcFramework.zip</p> <p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.15.0/ios</p> <ul style="list-style-type: none"> • Menambahkan <code>AVPictureInPictureController</code> ekstensi untuk memungkinkan pembuatan instance baru dengan <code>fileIVSImagePreviewView</code> . • Menambahkan API baru <code>IVSImageDeviceAVSampleBufferDisplayLayer</code> untuk membuat perangkat yang dirender.

Platform	Unduhan dan Perubahan
	<ul style="list-style-type: none"> • Memperbaiki masalah bitrate rendah pada perangkat yang menjalankan iOS 17 dan yang lebih baru. • Perbaiki bug minor.

Ukuran SDK Siaran: Android

Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64-v8a	5,243 MB	13.194 MB
armeabi-v7a	4,541 MB	9.188 MB
x86_64	5,628 MB	14.455 MB
x86	5,434 MB	14.046 MB

Ukuran SDK Siaran: iOS

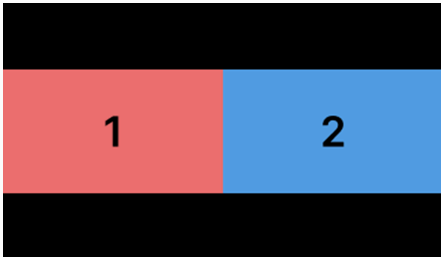
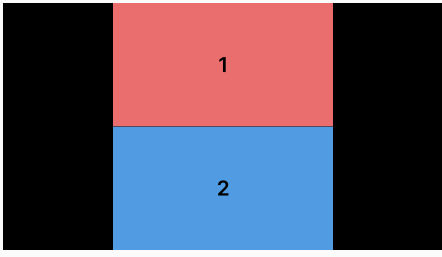
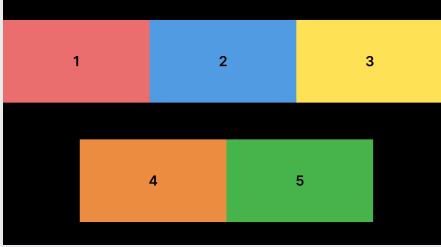
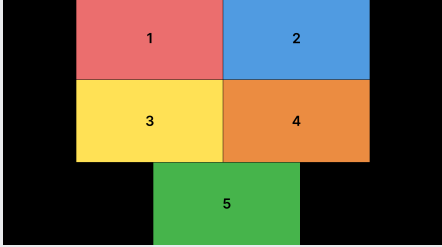
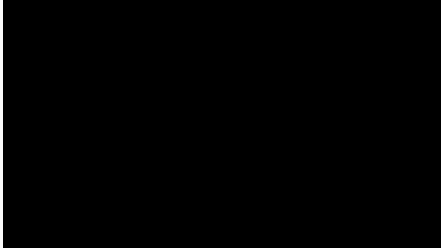
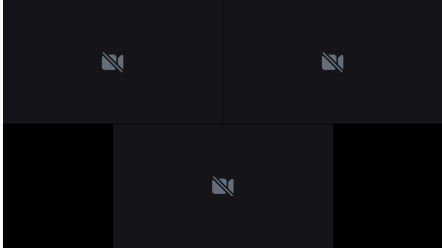
Arsitektur	Ukuran Terkompresi	Ukuran Tidak Terkompresi
arm64	3,358 MB	7,820 MB

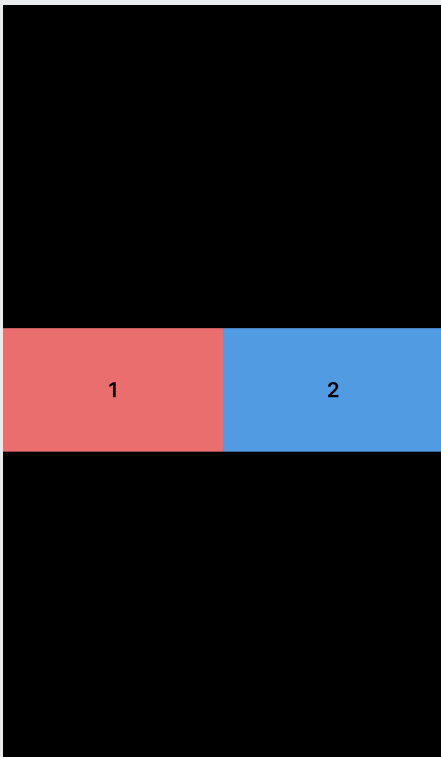
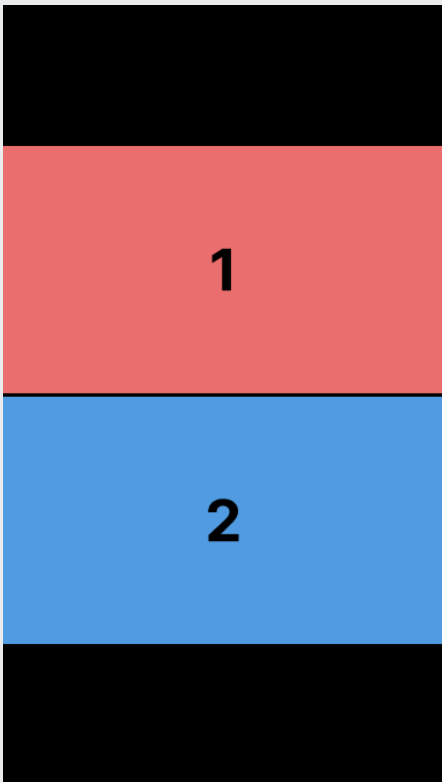
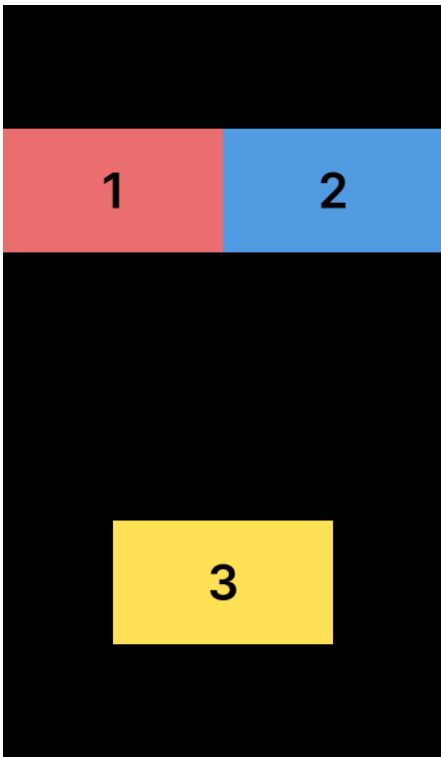
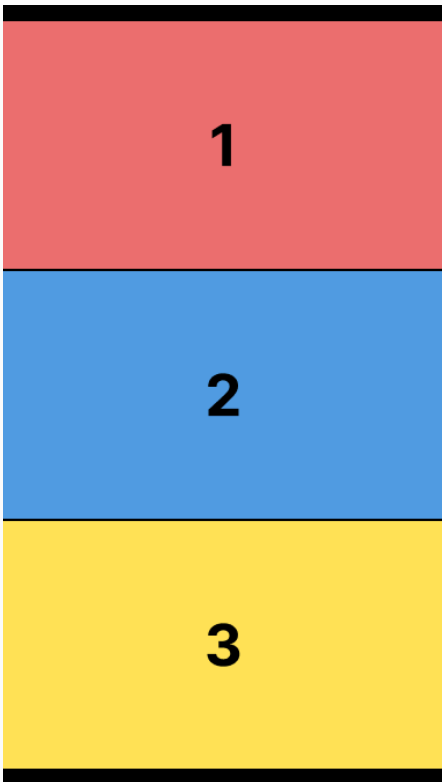
Februari 7, 2024

Pembaruan Tata Letak Komposisi Sisi Server

Rilis ini memperkenalkan perbaikan visual pada tata letak grid default. Perubahan ini akan mengoptimalkan bagaimana video ditampilkan dan mengurangi ruang kosong. Perubahan ini akan diaktifkan pada 7 Maret 2024.

Penting: Pastikan aplikasi Anda tidak bergantung pada fitur spesifik dari tata letak saat ini, seperti ukuran dan posisi ubin. Perbaikan visual pada tata letak dapat diperkenalkan kapan saja.

Deskripsi Perubahan	Tua	Baru
<p>Secara otomatis memilih penempatan peserta yang optimal untuk memaksimalkan ukuran video.</p>		
<p>Meningkatkan pemanfaatan ruang dengan mengurangi kesenjangan dan meminimalkan bilah hitam.</p>		
<p>Menambahkan indikator “kamera mati” baru untuk visibilitas yang jelas dari peserta yang tidak berbagi video.</p>		

Deskripsi Perubahan	Tua	Baru
<p>Meningkatkan pemanfaatan ruang dan proporsi untuk kasus penggunaan potret.</p>		
<p>Meningkatkan pemanfaatan ruang dalam kasus penggunaan potret dengan meminimalkan jarak antar peserta dan mengurangi letterboxing atau pillarboxing.</p>		

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-referensi</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.
SDK Siaran Android 1.14.1	<p>Dokumentasi referensi: https://aws.github.io/amazon-ivs-broadcast-docs/1.14.1/android</p>

Platform	Unduhan dan Perubahan
	<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 libWebRTC 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-referensi</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 penerbitan. enable/disable
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 <code>attributes</code> ke <code>ParticipantInfo</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 and/or yang digunakan <code>fpsStageVideoConfiguration # setCameraCaptureQuality</code>. • <code>IllegalArgumentException</code> dilemparkan saat mengatur properti konfigurasi sekarang menyertakan nilai yang disediakan dalam pesan pengecualian.

Platform	Unduhan dan Perubahan
SDK Siaran iOS 1.13.0	<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 <code>libWebRTC</code> 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 kami SDKs sekarang memiliki urutan penyortiran yang dijamin, seperti yang didokumentasikan dalam SDKs diri mereka 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. Untuk informasi lebih lanjut, lihat:

- [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. Untuk informasi lebih lanjut, lihat:

- [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-referensi</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-referensi</p> <ul style="list-style-type: none"> Memperbaiki bug yang mencegah penerbitan ulang <code>refreshStrategy</code> saat status yang dipublikasikan 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-referensi</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 yang 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

Platform	Unduhan dan Perubahan
	<p>untuk mendeteksi kapan frame pertama diberikan untuk peserta jarak jauh dalam suatu tahap.</p>
<p>SDK Siaran Android 1.12.0</p>	<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.
<p>SDK Siaran iOS 1.12.0</p>	<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-referensi</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.

Rilis ini disertai perubahan besar pada dokumentasi. [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.