



Guía del usuario de transmisión en tiempo real

Amazon IVS



Amazon IVS: Guía del usuario de transmisión en tiempo real

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

Las marcas comerciales y la imagen comercial de Amazon no se pueden utilizar en relación con ningún producto o servicio que no sea de Amazon de ninguna manera que pueda causar confusión entre los clientes y que menosprecie o desacredite a Amazon. Todas las demás marcas registradas que no son propiedad de Amazon son propiedad de sus respectivos propietarios, que pueden o no estar afiliados, conectados o patrocinados por Amazon.

Table of Contents

¿Qué es el streaming en tiempo real de IVS?	1
Solución global, control regional	2
La transmisión y la visualización son globales	2
El control es regional	2
Primeros pasos con IVS	4
Introducción	4
Requisitos previos	4
Otras referencias	4
Terminología de transmisión en tiempo real	5
Información general sobre los pasos	5
Paso 1: configurar permisos de IAM	6
Uso de una política existente para los permisos de IVS	6
Opcional: crear una política personalizada para los permisos de Amazon IVS	7
Crear usuarios nuevos y agregar permisos	8
Agregar permisos para un usuario existente	9
Paso 2: crear una fase	10
Instrucciones de la consola	10
Instrucciones de la CLI	11
Paso 3: distribuir los tokens de participantes	12
Creación de tokens con un par de claves	12
Creación de tokens con la API de transmisión en tiempo real de IVS	18
Paso 4: integrar el SDK de transmisión de IVS	20
Web	21
Android	22
iOS	23
Paso 5: publicar y suscribirse a videos	24
Consola de IVS	24
Web	25
Android	33
iOS	58
Supervisión	89
¿Qué es la sesión de una fase?	89
Ver sesiones de fases y participantes	89
Instrucciones de la consola	89

Ver eventos de un participante	89
Instrucciones de la consola	90
Instrucciones de la CLI	90
Acceso a métricas de CloudWatch	91
Instrucciones de la consola de CloudWatch	91
Instrucciones de la CLI	92
Métricas de CloudWatch: streaming en tiempo real de IVS	92
SDK de transmisión de IVS	98
Requisitos de la plataforma	99
Plataformas nativas	99
Navegadores de escritorio	99
Navegadores móviles (iOS y Android)	100
Vistas web	100
Se requiere acceso a los dispositivos	100
Soporte	101
Control de versiones	101
Guía web	102
Introducción	103
Publicación y suscripción	105
Problemas conocidos y soluciones alternativas	118
Control de errores	121
Guía de Android	124
Introducción	125
Publicación y suscripción	128
Problemas conocidos y soluciones alternativas	140
Control de errores	141
Guía para iOS	144
Introducción	145
Publicación y suscripción	147
Cómo elige iOS la resolución de la cámara y la velocidad de fotogramas	157
Problemas conocidos y soluciones alternativas	159
Control de errores	160
Fuentes de imágenes personalizadas	163
Android	163
iOS	164
Filtros de cámara externos	165

Integración de filtros de cámara externos	165
BytePlus	166
DeepAR	167
Snap	168
Reemplazo de fondo	183
Modos de audio móvil	205
Introducción	205
Modo preestablecido de audio	206
Casos de uso avanzados	209
Integración con otros servicios de SDK	211
Uso de Amazon EventBridge con IVS	213
Creación de reglas de Amazon EventBridge para Amazon IVS	215
Ejemplos: cambio del estado de la composición	216
Ejemplos: cambio en el estado de grabación de un participante individual	219
Ejemplos: actualización de una fase	222
Composición del servidor	224
Información general	224
Ventajas	225
Vida útil de la Composición	226
API de IVS	227
Diseños	228
Introducción	230
Requisitos previos	231
Instrucciones de la CLI	232
Habilitación de la pantalla compartida	234
Creación del recurso de EncoderConfiguration	234
Inicio de la Composición	235
Detención de la composición	237
Registro	239
Grabación de participantes individuales	239
Grabación compuesta	240
Grabación de participantes individuales	240
Introducción	240
Flujo de trabajo	241
Grabación solo de audio	244
Grabación de contenidos	245

Archivos de metadatos JSON	246
Grabación compuesta	251
.....	240
Grabación de contenidos	254
Política de bucket para StorageConfiguration	255
Archivos de metadatos JSON	256
Reproducción de contenido grabado desde buckets privados	261
Resolución de problemas	266
Problema conocido	266
Ingesta de transmisiones	267
Protocolos admitidos	267
Especificaciones de contenido multimedia compatible	268
RTMP	269
Creación de una fase	269
Creación de una configuración de ingesta	269
Publicación con un codificador RTMP	270
WHIP	271
Guía para OBS	271
Service Quotas	273
Aumentos en la cuota de servicio	273
Cuotas de tarifa de llamadas a la API	273
.....	273
Otras cuotas	275
.....	275
Optimizaciones de transmisión	278
Introducción	278
Transmisión adaptativa: codificación en capas con transmisión simultánea	278
Capas, calidades y velocidades de fotogramas predeterminadas	279
Resolución de las capas	280
Configuración de la codificación en capas con transmisión simultánea	281
Configuraciones de transmisión	281
Cambiar la velocidad de bits de la transmisión	282
Cambiar la velocidad de fotogramas de la transmisión de video	282
Optimización de la tasa de bits de audio y el soporte estéreo	284
Cambio en el valor de MinDelay del búfer de fluctuación del suscriptor	285
Optimizaciones sugeridas	286

Recursos y soporte	288
Recursos	288
Demostraciones	288
Soporte	289
Glosario	290
Historial de documentos	312
Cambios en la Guía del usuario de transmisión en tiempo real	312
Cambios en la Referencia de la API de transmisión en tiempo real de IVS	338
Notas de la versión	342
10 de octubre de 2024	342
SDK de transmisión de IVS: versión 1.17.0 para web (transmisión en tiempo real)	342
10 de octubre de 2024	342
SDK de transmisión de Amazon IVS: versión 1.23.0 para Android y versión 1.23.0 para iOS (transmisión en tiempo real)	342
11 de septiembre de 2024	344
SDK de transmisión de Amazon IVS: versión 1.22.0 para Android y versión 1.22.0 para iOS (transmisión en tiempo real)	344
11 de septiembre de 2024	345
SDK de transmisión de IVS: versión 1.16.0 para web (transmisión en tiempo real)	345
9 de septiembre de 2024	345
Ingesta RTMP	345
19 de agosto de 2024	345
Publicación o suscripción en la consola	345
15 de agosto de 2024	346
SDK de transmisión de IVS: versión 1.15.0 para web (transmisión en tiempo real)	346
15 de agosto de 2024	347
SDK de transmisión de Amazon IVS: versión 1.21.0 para Android y versión 1.21.0 para iOS (transmisión en tiempo real)	347
18 de julio de 2024	349
SDK de transmisión de IVS: versión 1.14.0 para web (transmisión en tiempo real)	349
18 de julio de 2024	349
SDK de transmisión de Amazon IVS: versión 1.20.0 para Android y versión 1.20.0 para iOS (transmisión en tiempo real)	349
26 de junio de 2024	350
Generación de tokens de participantes con un par de claves	350
20 de junio de 2024	351

Grabación de participantes individuales	351
13 de junio de 2024	351
SDK de transmisión de Amazon IVS: versión 1.19.0 para Android y versión 1.19.0 para iOS (transmisión en tiempo real)	351
13 de junio de 2024	352
SDK de transmisión de IVS: versión 1.13.0 para web (transmisión en tiempo real)	352
20 de mayo de 2024	353
SDK de transmisión de IVS: versión 1.12.0 para web (transmisión en tiempo real)	353
16 de mayo de 2024	354
SDK de transmisión de Amazon IVS: versión 1.18.0 para Android y versión 1.18.0 para iOS (transmisión en tiempo real)	354
6 de mayo de 2024	355
SDK de transmisión de IVS: versión 1.11.0 para web (transmisión en tiempo real)	355
30 de abril de 2024	356
SDK de transmisión de IVS: versión 1.10.1 para web (transmisión en tiempo real)	356
30 de abril de 2024	356
SDK de transmisión de Amazon IVS: versión 1.15.2 para Android y versión 1.15.2 iOS (transmisión en tiempo real)	356
22 de abril de 2024	357
SDK de transmisión de Amazon IVS: versión 1.17.0 para Android y versión 1.17.0 para iOS (transmisión en tiempo real)	357
21 de marzo de 2024	359
SDK de transmisión de Amazon IVS: versión 1.16.0 para Android, versión 1.16.0 para iOS y versión 1.10.0 para web (transmisión en tiempo real)	359
13 de marzo de 2024	360
SDK de transmisión de Amazon IVS: versión 1.15.1 para Android y versión 1.15.1 para iOS (transmisión en tiempo real)	360
13 de marzo de 2024	361
Actualizaciones de la API de composición del servidor	361
8 de marzo de 2024	362
Actualizaciones del diseño de composición del servidor	362
22 de febrero de 2024	362
SDK de transmisión de Amazon IVS: versión 1.15.0 para Android, versión 1.15.0 para iOS y versión 1.9.0 para web (transmisión en tiempo real)	362
7 de febrero de 2024	364
Actualizaciones del diseño de composición del servidor	364

6 de febrero de 2024	366
Compatibilidad con OBS y WHIP	366
1 de febrero de 2024	366
SDK de transmisión de Amazon IVS: versión 1.14.1 para Android, versión 1.14.1 para iOS y versión 1.8.0 para web (transmisión en tiempo real)	366
3 de enero de 2024	369
SDK de transmisión de Amazon IVS: versión 1.13.4 para Android, versión 1.13.4 para iOS y versión 1.7.0 para web (transmisión en tiempo real)	369
7 de diciembre de 2023	371
Nuevas métricas de CloudWatch	371
4 de diciembre de 2023	371
SDK de transmisión de Amazon IVS: Android 1.13.2 e iOS 1.13.2 (transmisión en tiempo real)	371
21 de noviembre de 2023	373
SDK de transmisión de Amazon IVS: Android 1.13.1 (transmisión en tiempo real)	373
17 de noviembre de 2023	373
SDK de transmisión de Amazon IVS: Android 1.13.0 e iOS 1.13.0 (transmisión en tiempo real)	373
16 de noviembre de 2023	378
Grabación compuesta	378
16 de noviembre de 2023	379
Composición del servidor	379
16 de octubre de 2023	380
SDK de transmisión de Amazon IVS: Web 1.6.0 (transmisión en tiempo real)	380
12 de octubre de 2023	380
Nuevas métricas y datos de los participantes de CloudWatch	380
12 de octubre de 2023	381
SDK de transmisión de Amazon IVS: Android 1.12.1 (transmisión en tiempo real)	381
14 de septiembre de 2023	382
SDK de transmisión de Amazon IVS: Web 1.5.2 (transmisión en tiempo real)	382
23 de agosto de 2023	382
SDK de transmisión de Amazon IVS: Web 1.5.1, Android 1.12.0 e iOS 1.12.0 (transmisión en tiempo real)	382
7 de agosto de 2023	385
SDK de transmisión de Amazon IVS: Web 1.5.0, Android 1.11.0 y iOS 1.11.0	385
7 de agosto de 2023	387

Transmisión en tiempo real 387

¿Qué es el streaming en tiempo real de Amazon IVS?

El streaming en tiempo real de Amazon Interactive Video Service (IVS) le ofrece todo lo que necesita para agregar audio y video en tiempo real a sus aplicaciones.

Ventajas:

- **Latencia en tiempo real:** cree aplicaciones para casos de uso sensibles a la latencia y ayude a sus espectadores a mantenerse conectados e interesados en el streaming en tiempo real de IVS. Ofrezca transmisiones en directo con una latencia inferior a 300 milisegundos desde el host hasta el espectador.
- **Alta simultaneidad:** libere el potencial de las interacciones a gran escala con el streaming en tiempo real de IVS. Atienda a audiencias de hasta 25 000 espectadores y permita que hasta 12 hosts suban al escenario virtual.
- **Optimizado para dispositivos móviles:** el streaming en tiempo real de IVS está optimizado para casos de uso móvil y se adapta a una amplia gama de dispositivos y capacidades de red. Al integrar los SDK de transmisión de Amazon IVS para Android y iOS, sus usuarios pueden interactuar como hosts o espectadores y disfrutar de transmisiones en directo de alta calidad en sus dispositivos móviles.

Casos de uso

- **Lugares para invitados:** cree aplicaciones que permitan a los hosts promocionar a los invitados “al escenario” y convertir a los espectadores en hosts para interactuar en tiempo real.
- **Modo Versus (VS):** cree experiencias con competiciones paralelas y permita a los espectadores ver a los hosts competir en tiempo real.
- **Salas de audio:** invite a los oyentes a unirse a la conversación como invitados y fomente una mayor participación en sus salas de audio.
- **Subastas de video en vivo:** convierta las subastas en eventos de video interactivos y mantenga su entusiasmo e integridad con latencia en tiempo real.

Además de la documentación del producto aquí, consulte <https://ivs.rocks/>, un sitio dedicado a explorar el contenido publicado (demostraciones, ejemplos de código, publicaciones de blog), calcular el costo y experimentar Amazon IVS a través de demostraciones en vivo.

Solución global, control regional

La transmisión y la visualización son globales

Puede utilizar Amazon IVS para transmitir a espectadores de todo el mundo:

- Cuando realiza una transmisión, Amazon IVS incorpora automáticamente el video en una ubicación cercana a usted.
- Los espectadores pueden ver sus transmisiones en directo en todo el mundo.

Otra forma de decir esto es que el "plano de datos" es global. El plano de datos se refiere al streaming/la incorporación y la visualización.

El control es regional

Aunque el plano de datos de Amazon IVS es global, el "plano de control" es regional. El plano de control hace referencia a la consola, la API y los recursos (escenarios) de Amazon IVS.

Otra forma de decir esto es que Amazon IVS es un "servicio de AWS regional". Es decir, los recursos de Amazon IVS en cada región son independientes de los recursos similares de otras regiones. Por ejemplo, un escenario que crea en una región es independiente de los escenarios que crea en otras regiones.

Cuando utilice recursos (por ejemplo, creación de un escenario), debe especificar la región en los que se crearán. Posteriormente, al administrar recursos, debe hacerlo desde la misma región en la que se crearon.

Si usa la...	Especifica la región...
Consola de Amazon IVS	Mediante el menú desplegable Select a Region (Seleccionar una región) en la parte superior derecha de la barra de navegación.
API de Amazon IVS	Mediante el punto de enlace de servicio correspondiente. Consulte la Referencia de la API de streaming en tiempo real de Amazon IVS . (Si accede a la API a través de un SDK, configure el parámetro <code>region</code> del SDK. Consulte Herramientas para crear en AWS).

Si usa la...	Especifica la región...
AWS CLI	Con cualquiera de las siguientes opciones: <ul style="list-style-type: none"><li data-bbox="472 306 1398 342">• Agregar la <code>--region <aws-region></code> al comando de la CLI.<li data-bbox="472 363 1386 399">• Colocar la región en el archivo de configuración local de AWS.

Recuerde que, independientemente de la región en la que se haya creado un escenario, puede transmitir a Amazon IVS desde cualquier lugar y los espectadores pueden verlo desde cualquier lugar.

Introducción a transmisión en tiempo real de IVS

Este documento explica los pasos necesarios para integrar la transmisión en tiempo real de Amazon IVS en su aplicación.

Temas

- [Introducción a la transmisión en tiempo real de IVS](#)
- [Paso 1: configurar permisos de IAM](#)
- [Paso 2: crear una fase](#)
- [Paso 3: distribuir los tokens de participantes](#)
- [Paso 4: integrar el SDK de transmisión de IVS](#)
- [Paso 5: publicar y suscribirse a videos](#)

Introducción a la transmisión en tiempo real de IVS

En esta sección se enumeran los requisitos previos para utilizar la transmisión en tiempo real y se presenta terminología clave.

Requisitos previos

Antes de usar la transmisión en tiempo real por primera vez, lleve a cabo las siguientes tareas: Para obtener instrucciones, consulte [Introducción a transmisión de baja latencia de IVS](#).

- Cree una cuenta de AWS
- Configuración de usuarios raíz y administrativos

Otras referencias

- [Referencia del SDK de transmisión web de IVS](#)
- [Referencia del SDK de transmisión para Android de IVS](#)
- [Referencia del SDK de transmisión para iOS de IVS](#)
- [Referencia de la API de transmisión en tiempo real de IVS](#)

Terminología de transmisión en tiempo real

Plazo	Descripción
Escenario	Un espacio virtual donde los participantes pueden intercambiar videos en tiempo real.
Host	Un participante que envía un video de un entorno local al escenario.
Visor	Un participante que recibe un video de los hosts.
Participante	Un usuario conectado al escenario como host o espectador.
Token de participante	Un token que autentica a un participante cuando se une a un escenario.
SDK de transmisión	Una biblioteca cliente que permite a los participantes enviar y recibir videos.

Información general sobre los pasos

1. [Configuración de permisos de IAM](#): cree una política de AWS Identity and Access Management (IAM) que proporcione a los usuarios un conjunto básico de permisos y asigne esa política a los usuarios.
2. [Creación de un escenario](#): cree un espacio virtual donde los participantes pueden intercambiar videos en tiempo real.
3. [Distribución de los tokens de participante](#): envíe tokens a los participantes para que puedan unirse a su escenario.

4. [Integración del SDK de transmisión de IVS](#): agregue el SDK de transmisión a su aplicación para permitir a los participantes enviar y recibir videos: [the section called “Web”](#), [the section called “Android”](#) y [the section called “iOS”](#).
5. [Publicación de videos y suscripción para recibirlos](#): envíe su video a la fase y reciba videos de otros hosts: [consola de IVS](#), [the section called “Web”](#), [the section called “Android”](#) y [the section called “iOS”](#).

Paso 1: configurar permisos de IAM

A continuación, debe crear una política de AWS Identity and Access Management (IAM) que proporcione a los usuarios una serie de permisos básicos (por ejemplo, para crear un escenario Amazon IVS y crear tokens del participante), y asignar dicha política a los usuarios. Puede asignar los permisos cuando crea un [usuario nuevo](#) o agregarlos a un [usuario actual](#). A continuación, se explican ambos procedimientos.

Para obtener más información (por ejemplo, para obtener información sobre los usuarios y las políticas de IAM, cómo adjuntar una política a un usuario y cómo restringir lo que los usuarios pueden hacer con Amazon IVS), consulte:

- [Crear un usuario de IAM](#) en la Guía del usuario de IAM.
- La información de [Seguridad de Amazon IVS](#) sobre IAM y “Managed Policies for IVS”.
- La información de IAM en [Seguridad en Amazon IVS](#)

Puede utilizar una política administrada de AWS existente para Amazon IVS o crear una nueva que personalice los permisos que quiera conceder a un conjunto de usuarios, grupos o roles. A continuación se describen ambos enfoques.

Uso de una política existente para los permisos de IVS

En la mayoría de los casos, querrá utilizar una política administrada de AWS para Amazon IVS. Se describen detalladamente en la sección [Managed Policies for IVS](#) de Seguridad de IVS.

- Utilice la política administrada de AWS `IVSReadOnlyAccess` para ofrecer a los desarrolladores de aplicaciones acceso a todos los puntos de conexión de las API Get y List de IVS (para transmisión de baja latencia y en tiempo real).

- Utilice la política administrada de AWS `IVSFullAccess` para ofrecer a los desarrolladores de aplicaciones acceso a todos los puntos de conexión de la API de IVS (para transmisión de baja latencia y en tiempo real).

Opcional: crear una política personalizada para los permisos de Amazon IVS

Siga estos pasos:

1. Inicie sesión en la consola de administración de AWS y abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. En el panel de navegación, seleccione Políticas y, a continuación, elija Crear política. Se abre la ventana Especificar permisos.
3. En la ventana Especificar permisos, elija la pestaña JSON. Luego, copie y pegue la siguiente política de IVS en el área de texto del Editor de políticas. (La política no incluye todas las acciones de Amazon IVS. Puede agregar o eliminar, es decir, permitir o denegar, los permisos de acceso a los puntos de conexión según sea necesario. Consulte la [referencia de la API de transmisión en tiempo real de IVS](#) para obtener más información sobre los puntos de conexión de IVS.)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ivs:CreateStage",
        "ivs:CreateParticipantToken",
        "ivs:GetStage",
        "ivs:GetStageSession",
        "ivs:ListStages",
        "ivs:ListStageSessions",
        "ivs:CreateEncoderConfiguration",
        "ivs:GetEncoderConfiguration",
        "ivs:ListEncoderConfigurations",
        "ivs:GetComposition",
        "ivs:ListCompositions",
        "ivs:StartComposition",
        "ivs:StopComposition"
      ]
    }
  ],
}
```

```

    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "cloudwatch:DescribeAlarms",
      "cloudwatch:GetMetricData",
      "s3:DeleteBucketPolicy",
      "s3:GetBucketLocation",
      "s3:GetBucketPolicy",
      "s3:PutBucketPolicy",
      "servicequotas:ListAWSDefaultServiceQuotas",
      "servicequotas:ListRequestedServiceQuotaChangeHistoryByQuota",
      "servicequotas:ListServiceQuotas",
      "servicequotas:ListServices",
      "servicequotas:ListTagsForResource"
    ],
    "Resource": "*"
  }
]
}

```

4. En la ventana Especificar permisos, elija Siguiente (desplácese a la parte inferior de la ventana para verlo). Se abre la ventana Revisar y crear.
5. En la ventana Revisar y crear, asígnele un nombre a la política y, si lo desea, agregue una descripción. Anote el nombre de la política, ya que lo necesitará cuando cree usuarios (más adelante). Elija Create policy (Crear política) (en la parte inferior de la ventana).
6. Volverá a la ventana de la consola de IAM, donde debería ver un banner que confirma la creación de la política nueva.

Crear usuarios nuevos y agregar permisos

Claves de acceso de usuario de IAM

Las claves de acceso de IAM constan de un ID de clave de acceso y de una clave de acceso secreta. Se utilizan para firmar las solicitudes programáticas que realiza a AWS. Si no tiene claves de acceso, puede crearlas mediante la consola de administración de AWS. Como práctica recomendada, no cree claves de acceso del usuario raíz.

El único momento que puede ver o descargar la clave de acceso secreta es cuando crea las claves de acceso. No puede recuperarla más adelante. Sin embargo, puede crear claves de acceso nuevas en cualquier momento; debe tener permisos para realizar las acciones de IAM requeridas.

Siempre almacene las claves de acceso de forma segura. Nunca las comparta con terceros (incluso si parece que la consulta proviene de Amazon). Para obtener más información, consulte [Administración de claves de acceso para usuarios de IAM](#) en la Guía del usuario de IAM.

Procedimiento

Siga estos pasos:

1. En el panel de navegación, elija Usuarios y la opción Agregar usuario. Se abre la ventana Especificar los detalles del usuario.
2. En la ventana Especificar los detalles del usuario:
 - a. En Detalles del usuario, escriba el nuevo nombre de usuario que se va a crear.
 - b. Active la casilla de verificación Acceso de usuario a la consola de administración de AWS.
 - c. En Contraseña de la consola, seleccione Contraseña generada de manera automática.
 - d. Seleccione la casilla de verificación El usuario debe crear una contraseña nueva en el siguiente inicio de sesión.
 - e. Elija Siguiente. Se abre la ventana Establecer permisos.
3. En Establecer permisos, elija Asociar directamente las políticas existentes. Se abre la ventana Políticas de permisos.
4. En el cuadro de búsqueda, ingrese el nombre de una política de IVS (ya sea una política administrada de AWS o una política personalizada creada con anterioridad). Cuando la encuentre, marque la casilla para seleccionar la política.
5. Elija Siguiente (en la parte inferior de la ventana). Se abre la ventana Revisar y crear.
6. En la ventana Revisar y crear, confirme que toda la información del usuario sea correcta y, a continuación, elija Crear usuario (en la parte inferior de la ventana).
7. Se abre la ventana Recuperar la contraseña, que contiene los detalles de inicio de sesión de la consola. Guarde esta información de forma segura para consultarla en el futuro. Cuando haya terminado, elija Volver a la lista de usuarios.

Agregar permisos para un usuario existente

Siga estos pasos:

1. Inicie sesión en la consola de administración de AWS y abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. En el panel de navegación, elija Users (Usuarios) y, a continuación, elija un nombre de usuario existente para actualizarlo. (Haga clic en el nombre para elegirlo; no marque la casilla de selección.)
3. En la página Resumen, en la pestaña Permisos, elija Agregar permisos. Se abre la ventana Agregar permisos.
4. Seleccione Asociar directamente las políticas existentes. Se abre la ventana Políticas de permisos.
5. En el cuadro de búsqueda, ingrese el nombre de una política de IVS (ya sea una política administrada de AWS o una política personalizada creada con anterioridad). Cuando encuentre la política, marque la casilla para seleccionarla.
6. Elija Siguiente (en la parte inferior de la ventana). Se abre la ventana Revisión.
7. En la ventana de Revisión, selecciona Agregar permisos (en la parte inferior de la ventana).
8. En la página Summary (Resumen), confirme que se agregó la política de IVS.

Paso 2: crear una fase

Un escenario es un espacio virtual donde los participantes pueden intercambiar videos en tiempo real. Es el recurso fundamental de la API de transmisión en tiempo real. Puede crear un escenario con la consola o el punto de conexión CreateStage.

Le recomendamos que, siempre que sea posible, cree un nuevo escenario para cada sesión lógica y la elimine cuando termine, en lugar de conservar los escenarios antiguos para su posible reutilización. Si los recursos obsoletos (escenarios antiguos que no se van a reutilizar) no se eliminan, es probable que alcance el límite de escenarios más rápido.

Instrucciones de la consola


1. Abra la [consola de Amazon IVS](#).

(También puede acceder a la consola de Amazon IVS a través de la [consola de administración de AWS](#)).

2. En el panel de navegación de la izquierda, selecciona Fases y, a continuación, seleccione Crear fase. Aparece la ventana Crear fase.

Amazon IVS > Video > Stages > Create stage

Create stage [Info](#)

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

▶ How Amazon IVS stages work

Setup

Stage name – *optional*

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

▶ Tags [Info](#)

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

Cancel

Create stage

3. Si lo desea, ingrese un nombre para la fase. Seleccione Crear fase para crear la fase. Aparece la página de detalles de la nueva fase.

Instrucciones de la CLI

Para instalar la AWS CLI, consulte [Install or update to the latest version of the AWS CLI](#).

Ahora puede utilizar la CLI para crear y administrar recursos. La API de escenarios se encuentra en el espacio de nombres `ivs-realtime`. Por ejemplo, para crear un escenario:

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

La respuesta es:

```
{
```

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

Paso 3: distribuir los tokens de participantes

Ahora que tiene una fase, debe crear y distribuir tokens a los participantes para que puedan unirse a la fase y empezar a enviar y recibir videos. Existen dos enfoques para generar tokens:

- [Creación](#) de tokens con un par de claves.
- [Creación de tokens con la API de transmisión en tiempo real de IVS](#).

A continuación se describen estos dos enfoques.

Creación de tokens con un par de claves

Puede crear tokens en la aplicación de servidor y distribuirlos entre los participantes para que se unan a una fase. Tiene que generar un par de claves públicas o privadas ECDSA para firmar los JWT e importar la clave pública en IVS. Luego, IVS puede verificar los tokens al unirse a la fase.

IVS no ofrece caducidad de claves. Si la clave privada está comprometida, debe eliminar la clave pública anterior.

Creación de un nuevo par de claves

Existen varias formas de crear un par de claves. A continuación, damos dos ejemplos.

Para crear un par de claves nuevo en la consola, siga estos pasos:

1. Abra la [consola de Amazon IVS](#). Elija la región de su fase si aún no está en ella.
2. En el menú de navegación izquierdo, elija Transmisión en tiempo real > Claves públicas.
3. Elija Crear clave pública. Aparece el cuadro de diálogo Crear clave pública.
4. Siga las indicaciones y elija Create (Crear).
5. Amazon IVS genera un nuevo par de claves. La clave pública se importa como un recurso de clave pública y la clave privada se pone inmediatamente a disposición para su descarga. La clave pública también se puede descargar más adelante si es necesario.

Amazon IVS genera la clave en el lado del cliente y no almacena la clave privada. Asegúrese de guardar la clave; no podrá recuperarla más tarde.

Para crear un par de claves nuevo EC P384 con OpenSSL (es posible que tenga que instalar [OpenSSL](#) primero), siga los pasos a continuación. Este proceso le permite acceder tanto a la clave privada como a la pública. Solo necesita la clave pública si quiere probar la verificación de los tokens.

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

Ahora, importe la clave pública nueva con las instrucciones que se incluyen a continuación.

Importación de la clave pública

Una vez que tenga un par de claves, puede importar la clave pública a IVS. Nuestro sistema no necesita la clave privada, pero usted la utiliza para firmar los tokens.

Para importar una clave pública actual con la consola:

1. Abra la [consola de Amazon IVS](#). Elija la región de su fase si aún no está en ella.
2. En el menú de navegación izquierdo, elija Transmisión en tiempo real > Claves públicas.
3. Seleccione Importar. Aparece el cuadro de diálogo Importar clave pública.
4. Siga las indicaciones y elija Import (Importar).
5. Amazon IVS importa la clave pública y genera un recurso de clave de pública.

Para importar una clave pública actual con la CLI:

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

Puede omitir `--region <aws-region>` si la región se encuentra en su archivo de configuración local de AWS.

Aquí tiene un ejemplo de respuesta:

```
{
```

```

    "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": {}
    }
  }
}

```

Solicitud de API

```

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

```

Generación y firma del token

Para obtener detalles sobre cómo trabajar con JWT y las bibliotecas compatibles para firmar tokens, visite jwt.io. En la interfaz jwt.io, debes introducir la clave privada para firmar los tokens. La clave pública solo es necesaria si desea verificar los tokens.

Todos los JWT tienen tres campos: encabezado, carga y firma.

Los esquemas JSON para el encabezado y la carga útil del JWT se describen a continuación. Como alternativa, puede copiar un JSON de muestra de la consola de IVS. Para obtener el JSON de encabezado y carga útil desde la consola de IVS:

1. Abra la [consola de Amazon IVS](#). Elija la región de su fase si aún no está en ella.
2. En el menú de navegación izquierdo, elija Transmisión en tiempo real > Fases.
3. Seleccione la fase que desea usar. Seleccione Ver detalles.
4. En la sección Tokens de participantes, seleccione el menú desplegable situado junto a Crear token.
5. Seleccione Cree el encabezado y la carga útil del token.
6. Rellene el formulario y copie el encabezado y la carga útil de JWT que se muestran en la parte inferior de la ventana emergente.

Esquema del token: encabezado

El encabezado especifica lo siguiente:

- `alg` es el algoritmo de firma. Este es ES384, un algoritmo de firma ECDSA que utiliza el algoritmo hash SHA-384.
- `typ` es el tipo de token, JWT.
- `kid` es el ARN de la clave pública utilizada para firmar el token. Debe ser el mismo ARN devuelto por la solicitud a la API [GetPublicKey](#).

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

Esquema del token: carga útil

La carga útil contiene datos específicos de IVS. Todos los campos excepto `user_id` son obligatorios.

- `RegisteredClaims` en la especificación de JWT son notificaciones reservadas que deben proporcionarse para que el token de la fase sea válido:
 - `exp` (fecha de vencimiento) es una marca de tiempo UTC Unix del momento en que caduca el token. (Una marca de tiempo Unix es un valor numérico que representa los segundos desde 1970-01-01T00:00:00Z UTC hasta la fecha y hora UTC especificadas, sin tener en cuenta los segundos intercalares). El token se valida cuando el participante se une a una fase. IVS proporciona los tokens con un TTL predeterminado de 12 horas, que es el valor que recomendamos; se puede ampliar hasta un máximo de 14 días a partir del momento de su emisión (`iat`). Debe ser un valor de tipo entero.
 - `iat` (momento de la emisión) es una marca de tiempo UTC Unix del momento en que se emitió el JWT. (Consulte la nota para `exp` sobre las marcas de tiempo Unix). Debe ser un valor de tipo entero.
 - `jti` (ID de JWT) es el ID del participante que se utiliza para rastrear el participante al que se concede el token y para hacer referencia a él. Cada token debe tener un ID de participante único. Debe ser una cadena que distinga mayúsculas de minúsculas, con una longitud máxima

de 64 caracteres, que contenga solo caracteres alfanuméricos, guiones (-) y guiones bajos (_). No se permite utilizar ningún otro carácter especial.

- `user_id` es un nombre opcional asignado por el cliente para ayudar a identificar el token; se puede usar para vincular a un participante con un usuario en los propios sistemas del cliente. Debe coincidir con el campo `userId` de la solicitud a la API [CreateParticipantToken](#). Puede ser cualquier texto codificado en UTF-8 y es una cadena de hasta 128 caracteres. Este campo se expone a todos los participantes de la fase y no se debe utilizar en la información de identificación personal ni confidencial.
- `resource` es el ARN de la fase; p. ej., `arn:aws:ivs:us-east-1:123456789012:stage/oRmLNwuCeMlQ`.
- `topic` es el ID de la fase, que se puede extraer del ARN de la fase. Por ejemplo, si el ARN de la fase es `arn:aws:ivs:us-east-1:123456789012:stage/oRmLNwuCeMlQ`, el ID de la fase es `oRmLNwuCeMlQ`.
- `events_url` debe ser el punto de conexión del evento devuelto por la operación `CreateStage` o `GetStage`. Se recomienda almacenar en caché este valor en el momento de la creación de la fase; el valor se puede almacenar en caché durante un máximo de 14 días. Un valor de ejemplo es `wss://global.events.live-video.net`.
- `whip_url` debe ser el punto de conexión WHIP devuelto por la operación `CreateStage` o `GetStage`. Se recomienda almacenar en caché este valor en el momento de la creación de la fase; el valor se puede almacenar en caché durante un máximo de 14 días. Un valor de ejemplo es `https://453fdfd2ad24df.global-bm.whip.live-video.net`.
- `capabilities` especifica las capacidades del token; los valores válidos son `allow_publish` y `allow_subscribe`. Para los tokens de solo suscripción, establezca únicamente `allow_subscribe` en `true`.
- `attributes` es un campo opcional en el que puede especificar los atributos proporcionados por la aplicación para codificarlos en el token y adjuntarlos a una fase. Las claves y valores de asignación pueden contener texto codificado en UTF-8. La longitud máxima de este campo es de 1 KB en total. Este campo se expone a todos los participantes de la fase y no se debe utilizar en la información de identificación personal ni confidencial.
- `version` debe ser `1.0`.

```
{
  "exp": 1697322063,
  "iat": 1697149263,
  "jti": "Mx6clRRHODPy",
  "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"
}
```

Esquema del token: firma

Para crear la firma, utilice la clave privada con el algoritmo especificado en el encabezado (ES384) para firmar el encabezado codificado y la carga codificada.

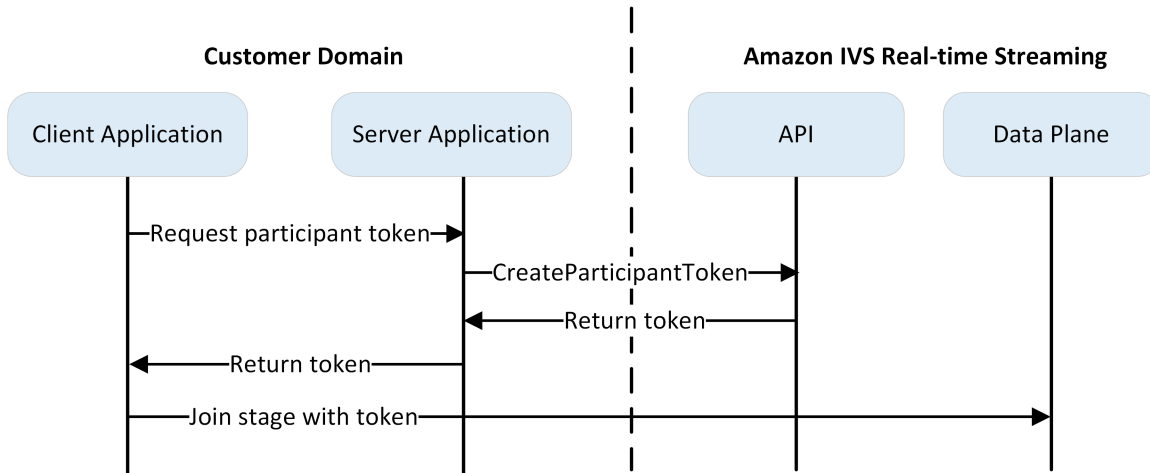
```
ECDSASHA384(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  <private-key>
)
```

Instrucciones

1. Para generar la firma del token con un algoritmo de firma ES384, firme el algoritmo y una clave privada asociada a la clave pública proporcionada a IVS.
2. Crear el token.

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

Creación de tokens con la API de transmisión en tiempo real de IVS



Como se muestra anteriormente, una aplicación cliente solicita un token a su aplicación del lado del servidor y esta llama a `CreateParticipantToken` mediante un SDK de AWS o una solicitud SigV4 firmada. Como las credenciales de AWS se utilizan para llamar a la API, el token debe generarse en una aplicación segura del lado del servidor, no en la aplicación del lado del cliente.

Al crear un token de participante, puede especificar opcionalmente atributos o capacidades:

- Puede especificar los atributos proporcionados por la aplicación para codificarlos en el token y adjuntarlos a una fase. Las claves y valores de asignación pueden contener texto codificado en UTF-8. La longitud máxima de este campo es de 1 KB en total. Este campo se expone a todos los participantes de la fase y no se debe utilizar en la información de identificación personal ni confidencial.
- Puede especificar las capacidades que habilita el token. El valor predeterminado es `PUBLISH` y `SUBSCRIBE`, que permite al participante enviar y recibir audio y video, pero puede emitir tokens con un subconjunto de capacidades. Por ejemplo, puede emitir un token que tenga únicamente la capacidad `SUBSCRIBE` para los moderadores. En ese caso, los moderadores podrían ver a los participantes que envían videos, pero no enviar los suyos propios.

Para obtener más información, consulte [CreateParticipantToken](#).

Puede crear tokens de participantes mediante la consola o la CLI para llevar a cabo pruebas y desarrollar, pero lo más probable es que le convenga más crearlos con el SDK de AWS en su entorno de producción.

Tendrá que distribuir los tokens de su servidor a cada cliente (por ejemplo, mediante una solicitud a la API). No ofrecemos esta funcionalidad. Para esta guía, basta con seguir estos pasos para copiar y pegar los tokens en el código del cliente:

Importante: trate los tokens como elementos opacos; es decir, no cree funciones en función del contenido del token. El formato de los tokens podría cambiar en el futuro.

Instrucciones de la consola

1. Vaya al escenario que creó en el paso anterior.
2. Seleccione Crear token. Aparece la ventana Crear token.
3. Ingrese un ID de usuario para asociarlo al token. Puede ser cualquier texto codificado en UTF-8.
4. Seleccione Crear.
5. Copie el token. Importante: Asegúrese de guardar el token; IVS no lo almacena y no podrá recuperarlo más adelante.

Instrucciones de la CLI

Crear un token con la AWS CLI requiere que antes descargue y configure la CLI en su equipo. Para obtener más información, consulte la [Guía del usuario de la interfaz de línea de comandos de AWS](#).

Nota: La generación de tokens con la AWS CLI es una buena opción para hacer pruebas, pero, para el uso en producción, le recomendamos que genere tokens en el lado del servidor con el SDK de AWS (consulte las instrucciones a continuación).

1. Ejecute el comando `create-participant-token` con el ARN del escenario. Incluya cualquiera de las siguientes capacidades: "PUBLISH", "SUBSCRIBE".

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

2. Esto devuelve un token de participante:

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


- Web: <https://codepen.io/amazon-ivs/pen/ZEqgrpo>
- Android: <https://github.com/aws-samples/amazon-ivs-real-time-transmisión-android-samples>
- iOS: <https://github.com/aws-samples/amazon-ivs-real-time-transmisión-ios-samples>

Web

Configuración de archivos

Para empezar, cree una carpeta y un archivo HTML y JS inicial para configurar los archivos:

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

Puede instalar el SDK de transmisión mediante una etiqueta script o npm. Nuestro ejemplo usa la etiqueta script por motivos de simplicidad, pero es fácil de modificar si opta por usar npm más adelante.

Uso de una etiqueta de script

El SDK de transmisión web se distribuye como biblioteca de JavaScript y se puede obtener en <https://web-broadcast.live-video.net/1.17.0/amazon-ivs-web-broadcast.js>.

Cuando se carga mediante una etiqueta `<script>`, la biblioteca muestra una variable global en el ámbito del intervalo denominado `IVSBroadcastClient`.

Con npm

Para instalar el paquete de npm:

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

Ahora puede acceder al objeto `IVSBroadcastClient`:

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

Android

Creación del proyecto para Android

1. Cree un nuevo proyecto en Android Studio.
2. Elija Actividad de vistas vacías.

Nota: En algunas versiones anteriores de Android Studio, la actividad basada en vistas se denomina Actividad vacía. Si aparece la ventana de Android Studio Actividad vacía y no Actividad de vistas vacías, seleccione Actividad vacía. De lo contrario, no seleccione Actividad vacía, ya que utilizaremos las API de vistas (no Jetpack Compose).

3. Asigne un nombre a su proyecto y, a continuación, seleccione Finalizar.

Instalación del SDK de transmisión

A fin de agregar la biblioteca de transmisión de Android de Amazon IVS a su entorno de desarrollo de Android, agregue la biblioteca al archivo `build.gradle` del módulo como se muestra a continuación (para la versión más reciente del SDK de transmisión de Amazon IVS). En los proyectos más recientes, es posible que el repositorio `mavenCentral` ya esté incluido en su archivo `settings.gradle`, si ese es el caso, puede omitir el bloqueo de `repositories`. Para nuestro ejemplo, también necesitaremos habilitar el enlace de datos en el bloque `android`.

```
android {
    dataBinding.enabled true
}

repositories {
    mavenCentral()
}

dependencies {
    implementation 'com.amazonaws:ivs-broadcast:1.23.0:stages@aar'
}
```

Alternativamente, para instalar el SDK de forma manual, descargue la última versión desde esta ubicación:

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

iOS

Creación del proyecto para iOS

1. Cree un nuevo proyecto de Xcode.
2. En Plataforma, seleccione iOS.
3. En Aplicación, seleccione Aplicación.
4. Ingrese el nombre del producto de su aplicación y, a continuación, seleccione Siguiente.
5. Para elegir un directorio en el que guardar el proyecto, vaya a este y, a continuación, seleccione Crear.

A continuación, tiene que incorporar el SDK. Recomendamos que integre el SDK de transmisión a través de CocoaPods. También puede agregar el marco a su proyecto de forma manual. Ambos métodos se describen a continuación.

Recomendación: integración del SDK de transmisión (CocoaPods)

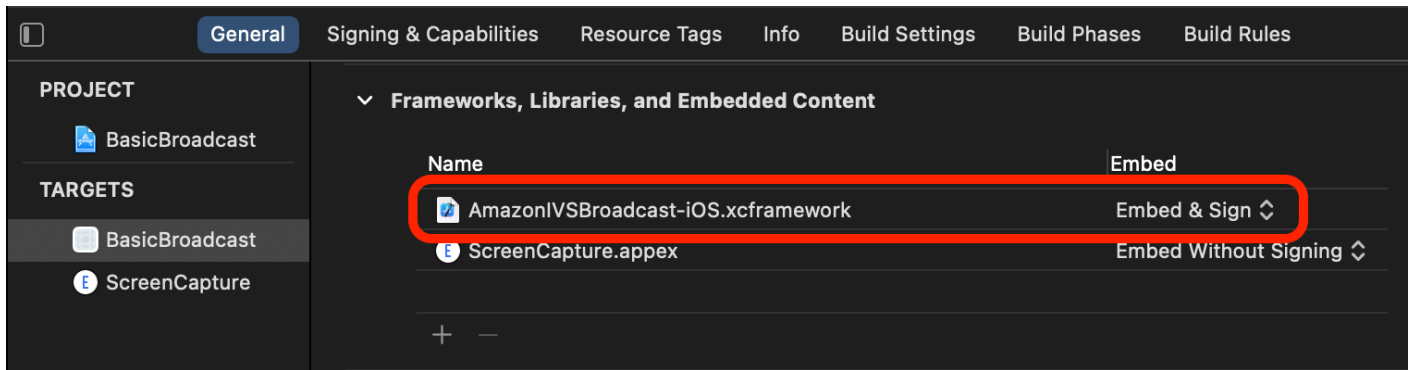
Supongamos que el nombre de su proyecto es `BasicRealTime`; cree un Podfile en la carpeta del proyecto con el siguiente contenido y, a continuación, ejecute `pod install`:

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

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

Método alternativo: instalar el marco de forma manual

1. Descargue la versión más reciente desde <https://broadcast.live-video.net/1.23.0/AmazonIVSBroadcast-Stages.xcframework.zip>.
2. Extraiga el contenido del archivo. `AmazonIVSBroadcast.xcframework` contiene el SDK para el dispositivo y el simulador.
3. Integre el `AmazonIVSBroadcast.xcframework` arrastrándolo a la sección Marcos, librerías y contenido integrado de la pestaña General para el destino de la aplicación:



Configuración de permisos

Tiene que actualizar el `Info.plist` de su proyecto para agregar dos entradas nuevas para `NSCameraUsageDescription` y `NSMicrophoneUsageDescription`. En los valores, proporcione explicaciones para el usuario sobre por qué la aplicación solicita acceso a la cámara y al micrófono.

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

Paso 5: publicar y suscribirse a videos

Puede publicar o suscribirse (en tiempo real) en IVS con:

- Los [SDK de transmisión nativos de IVS](#), que admiten WebRTC y RTMPS. Lo recomendamos, especialmente para escenarios de producción. Consulte los detalles que aparecen a continuación para la [web](#), [Android](#) y [iOS](#).
- La consola de Amazon IVS: es adecuada para probar transmisiones. Consulte a continuación.
- Otros codificadores de software y hardware de transmisión: puede usar cualquier codificador de transmisión que sea compatible con los protocolos RTMP, RTMPS o SRT. Para obtener más información, consulte [Ingesta de transmisiones](#).

Consola de IVS

1. Abra la [consola de Amazon IVS](#).

(También puede obtener acceso a la consola de Amazon IVS mediante la [consola de administración de AWS](#)).

2. En el panel de navegación, seleccione Fases. (Si el panel de navegación está contraído, amplíelo seleccionando el icono de la hamburguesa).
3. Seleccione la fase a la que desee suscribirse o en la que quiera publicar para ir a su página de detalles.
4. Suscripción: si la fase tiene uno o más publicadores, puede pulsar el botón Suscribirse, en la pestaña Suscribirse, para suscribirse a ella. (Las pestañas están debajo de la sección Configuración general).
5. Publicación:
 - a. Seleccione la pestaña Publicar.
 - b. Se le pedirá que conceda a la consola de IVS acceso a su cámara y micrófono; permita esos permisos.
 - c. En la parte inferior de la pestaña Publicar, utilice los cuadros desplegados para seleccionar los dispositivos de entrada para el micrófono y la cámara.
 - d. Para empezar a publicar, seleccione Empezar a publicar.
 - e. Para ver el contenido publicado, vuelva a la pestaña Suscribirse.
 - f. Para dejar de publicar, vaya a la pestaña Publicar y pulse el botón Dejar de publicar situado hacia la parte inferior.

Nota: La suscripción y la publicación consumen recursos y se le cobrará una tarifa por hora durante el tiempo que permanezca conectado a la fase. Para obtener más información, consulte [Streaming en tiempo real](#) en la página de precios de IVS.

Publicación y suscripción con el SDK de transmisión para web de IVS

En esta sección se explican los pasos necesarios para publicar y suscribirse a una fase mediante una aplicación web.

Creación de una plantilla HTML reutilizable

Primero, creemos la plantilla HTML reutilizable e importemos la biblioteca como una etiqueta script:

```
<!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/1.17.0/amazon-ivs-web-
broadcast.js"></script>
</head>

<body>

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

</body>
</html>
```

Configuración de la entrada de tokens y adición de botones de unión y salida

Aquí completamos el cuerpo con nuestros controles de entrada. Estos toman como entrada el token y configuran botones de unión y salida. Por lo general, las aplicaciones solicitarán el token a la API de la aplicación, pero, en este ejemplo, tiene que copiar y pegar el token en la entrada del 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 />
```

Adición de elementos de contenedor multimedia

Estos elementos albergarán el contenido multimedia para nuestros participantes locales y remotos. Agregamos una etiqueta script para cargar la lógica de nuestra aplicación definida en app.js.

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

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

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

Esto completa la página HTML y debería ver esto al cargar `index.html` en un navegador:

IVS Real-Time Streaming

Token

Join

Creación de `app.js`

Pasemos a definir el contenido de nuestro archivo `app.js`. Comience por importar todas las propiedades necesarias de la versión global del SDK:

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

Creación de las variables de la aplicación

Configure variables que contengan referencias a nuestros botones HTML de unión y salida y al estado de la aplicación:

```
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;
```

Creación de joinStage 1: definición de la función y validación de la entrada

La función `joinStage` toma el token de entrada, crea una conexión con el escenario y comienza a publicar el video y el audio recuperados de `getUserMedia`.

Para empezar, definimos la función y validamos el estado y la entrada del token. Vamos a desarrollar esta función en las próximas secciones.

```
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  
};
```

Creación de joinStage 2: obtención de contenido multimedia para su publicación

Este es el contenido multimedia que se publicará en el escenario:

```
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]);
```

Creación de joinStage 3: definición de la estrategia escénica y creación del escenario

Esta estrategia de escenarios es la base de la lógica de decisión que utiliza el SDK para decidir qué publicar y a qué participantes suscribirse. Para obtener más información sobre el propósito de la función, consulte [Estrategia](#).

Esta estrategia es sencilla. Cuando se una al escenario, publique las transmisiones que acabamos de recuperar y suscríbase al contenido de audio y video de todos los participantes remotos:

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

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

Creación de joinStage 4: gestión de los eventos del escenario y renderización del contenido multimedia

Los escenarios emiten muchos eventos. Tendremos que escuchar los eventos `STAGE_PARTICIPANT_STREAMS_ADDED` y `STAGE_PARTICIPANT_LEFT` para renderizar y

eliminar contenido multimedia hacia y desde la página. Puede encontrar un conjunto más exhaustivo de eventos en [Eventos](#).

Tenga en cuenta que creamos cuatro funciones auxiliares para poder administrar los elementos del DOM necesarios: `setupParticipant`, `teardownParticipant`, `createVideoEl` y `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;
  }
```

Creación de joinStage 5: unión al escenario

Vamos a unirnos por fin al escenario para completar la función `joinStage`.

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

Creación de leaveStage

Defina la función `leaveStage` que invocará el botón de salida.

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

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

Inicialización de los controladores de eventos de entrada

Agregaremos una última función al archivo `app.js`. Esta función se invoca inmediatamente cuando se carga la página y establece controladores de eventos para unirse al escenario y salir de este.

```
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
```

Ejecute la aplicación y proporcione un token

En este punto, puede compartir la página web de forma local o con otras personas, [abrir la página](#), introducir un token de participante y unirte al escenario.

Pasos siguientes

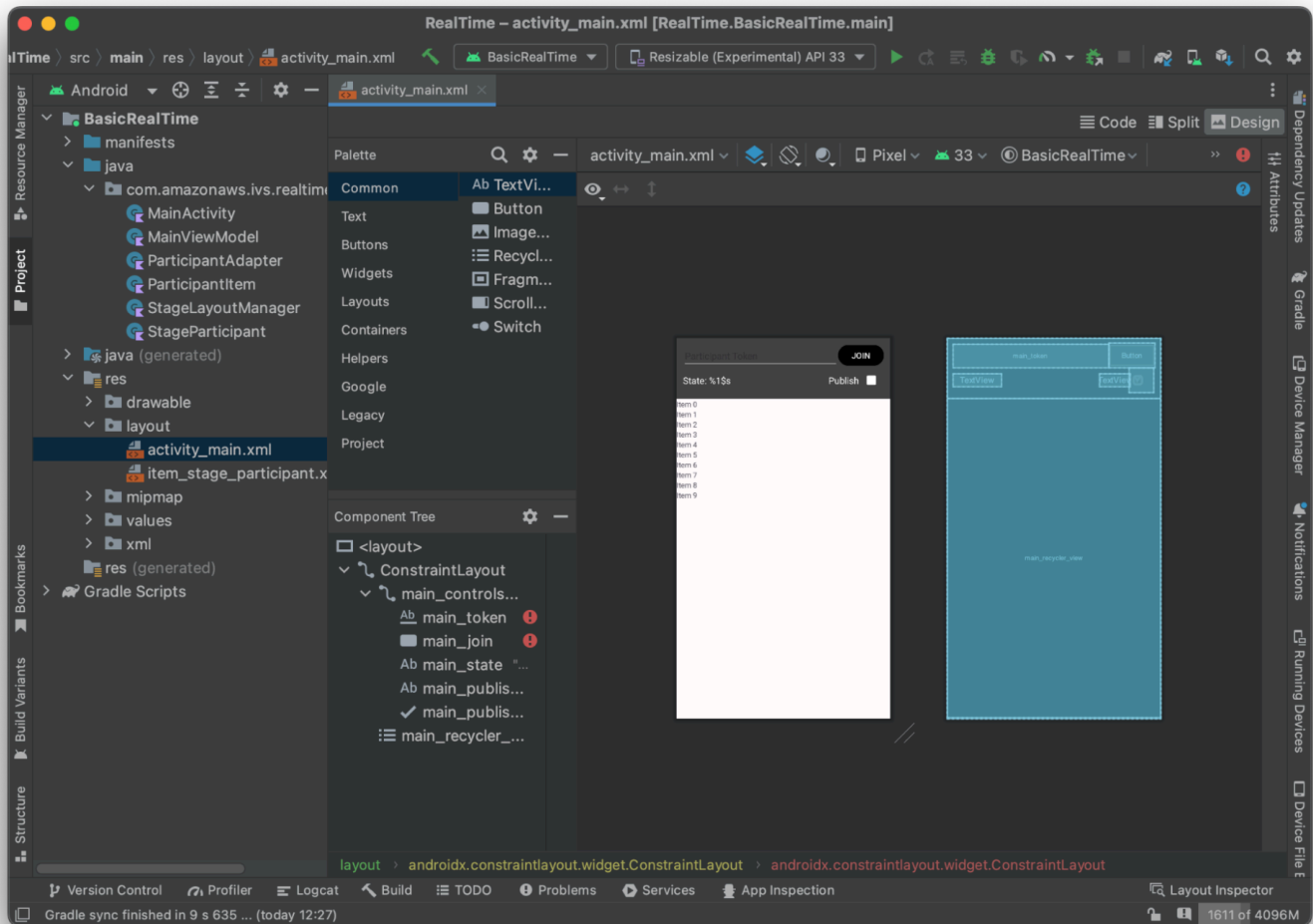
Para ver ejemplos más detallados sobre npm, React y otros, consulta la [Guía web del SDK de transmisión de IVS \(Guía de transmisión en tiempo real\)](#).

Publicación y suscripción con el SDK de transmisión para Android de IVS

En esta sección se explican los pasos necesarios para publicar y suscribirse a una fase mediante una aplicación Android.

Creación de vistas

Empezamos por crear un diseño simple para nuestra aplicación con el archivo `activity_main.xml` creado automáticamente. El diseño contiene `EditText` para agregar un token, un `Button` de unión, `TextView` para mostrar el estado del escenario y `CheckBox` para cambiar la publicación.



Este es el XML que hay detrás de la vista:

```
<?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>

```

Aquí hacemos referencia a los ID de un par de cadenas, así que es el momento de crear todo el archivo `strings.xml`:

```

<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>

```

Vamos a vincular esas vistas del XML a nuestro `MainActivity.kt`:

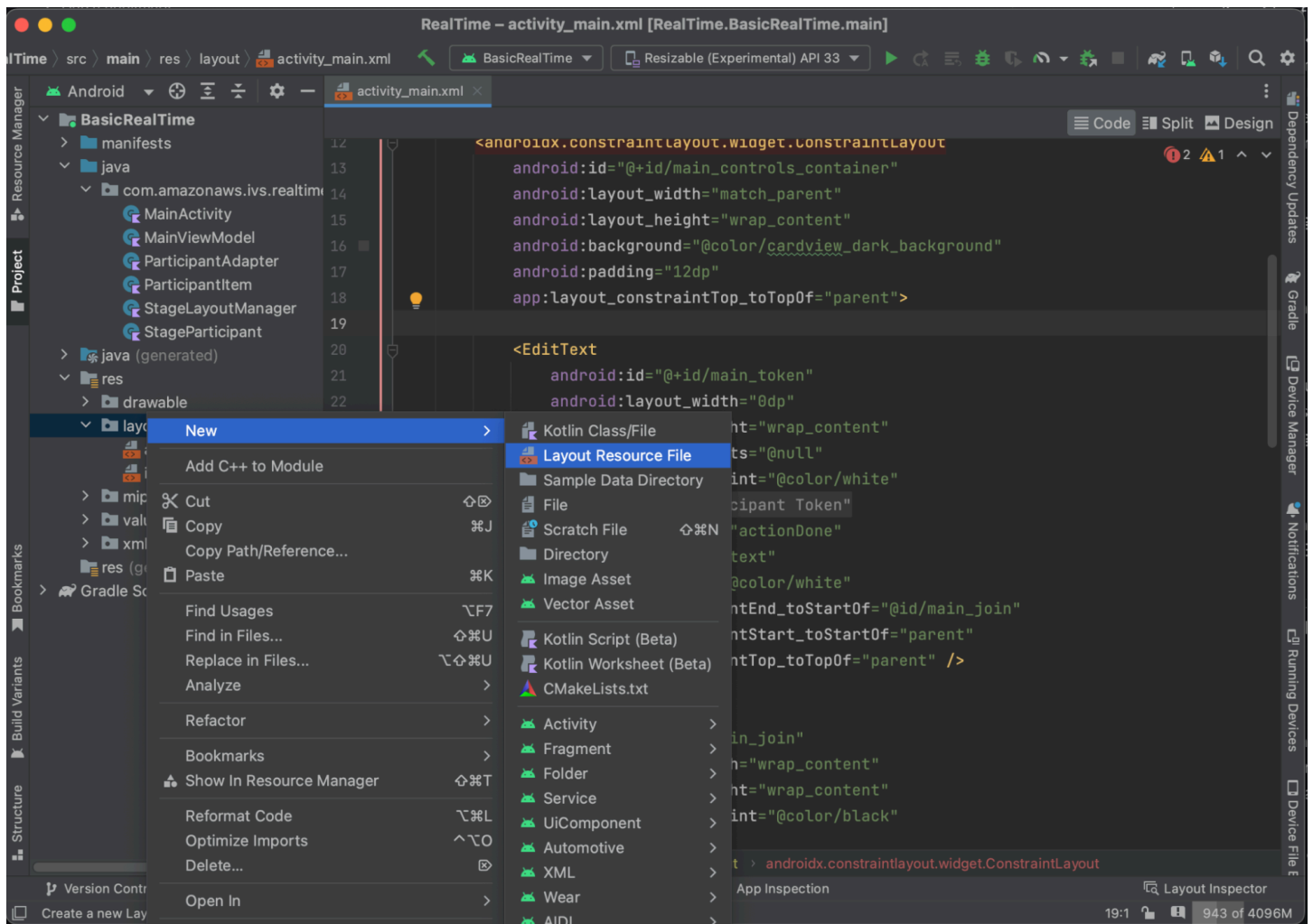
```
import android.widget.Button
import android.widget.CheckBox
import android.widget.EditText
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView

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

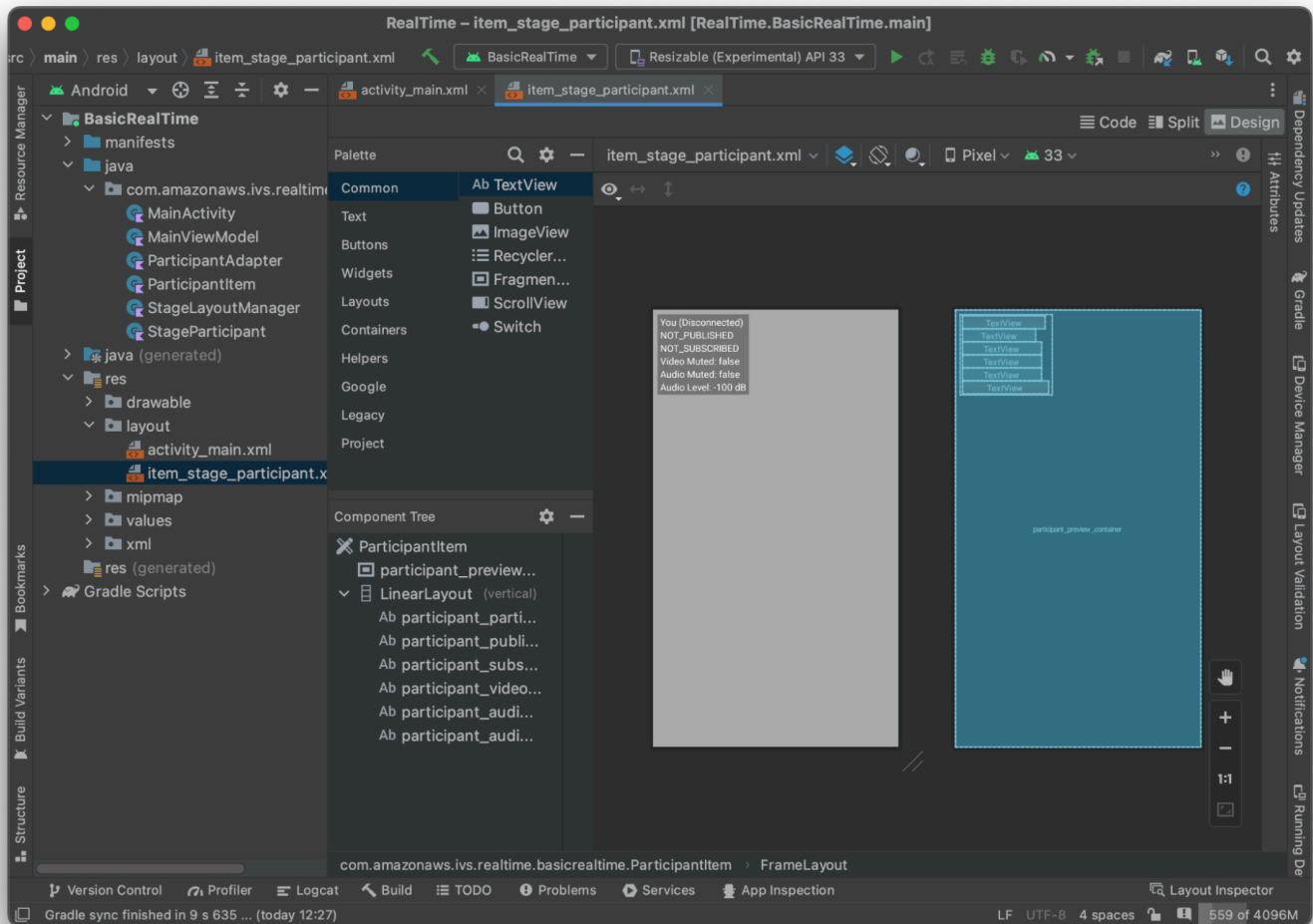
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    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)
}
```

Ahora crearemos una vista de elementos para `RecyclerView`. Para ello, haga clic con el botón derecho en el directorio `res/layout` y seleccione `Nuevo > Archivo de recursos de diseño`. Asigne a este archivo el nombre `item_stage_participant.xml`.



El diseño de este elemento es sencillo: contiene una vista para renderizar la transmisión de video de un participante y una lista de etiquetas para mostrar información sobre el participante:



Este es el XML:

```
<?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>
```

Este archivo XML infla una clase que aún no hemos creado, `ParticipantItem`. Como el XML incluye el espacio de nombres completo, asegúrese de actualizar este archivo XML en su espacio de nombres. Vamos a crear esta clase y configurar las vistas, aunque lo vamos a dejar en blanco por ahora.

Cree una nueva clase de Kotlin, `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)
}
}
```

Permisos

Para utilizar la cámara y el micrófono, debe solicitar permisos al usuario. Para ello, seguimos un flujo de permisos estándar:

```
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
    }
}
```

Estado de la aplicación

Nuestra aplicación hace un seguimiento de los participantes en el entorno local en un archivo `MainViewModel.kt`. El estado se comunicará de nuevo a `MainActivity` mediante [StateFlow](#) de Kotlin.

Cree una nueva clase de Kotlin (`MainViewModel`):

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

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

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

}
```

En `MainActivity.kt`, administramos el modelo de la vista:

```
import androidx.activity.viewModels

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

Para usar `AndroidViewModel` y las extensiones `ViewModel` de Kotlin, tendrá que agregar lo siguiente al archivo `build.gradle` del módulo:

```
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'
```

Adaptador RecyclerView

Crearemos una subclase `RecyclerView.Adapter` sencilla para hacer un seguimiento de nuestros participantes y actualizar `RecyclerView` en los eventos del escenario. Pero antes, necesitamos una clase que represente a un participante. Cree una nueva clase de Kotlin (`StageParticipant`):

```
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)
            }
        }
}
```

Usaremos esta clase en la clase `ParticipantAdapter` que crearemos a continuación.

Empezamos por definir la clase y crear una variable para hacer un seguimiento de los participantes:

```
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>()
```

También tenemos que definir `RecyclerView.ViewHolder` antes de implementar el resto de las anulaciones:

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

Con esto, podemos implementar las anulaciones de `RecyclerView.Adapter` estándar:

```
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)
    }
}

```

Por último, agregamos nuevos métodos a los que llamaremos desde `MainViewModel` cuando se hagan cambios en los participantes. Estos métodos son operaciones CRUD estándar en el adaptador.

```

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])
    }
}

```

En `MainViewModel`, tenemos que crear y mantener una referencia a este adaptador:

```
internal val participantAdapter = ParticipantAdapter()
```

Estado de la etapa

También tenemos que hacer un seguimiento del estado de algunos escenarios en `MainViewModel`. Definamos esas propiedades ahora:

```
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>()

```

Para ver su propia vista previa antes de unirse a un escenario, creamos inmediatamente un participante local:

```

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)
}

```

Queremos asegurarnos de borrar estos recursos cuando se elimine `ViewModel`. Anulamos `onCleared()` directamente para así no olvidarnos de eliminar estos recursos.

```

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

```

A continuación, completamos la propiedad `streams` local nada más recibir los permisos e implementamos el método `permissionsGranted` al que llamamos anteriormente:

```

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)
}
}

```

Implementación del SDK de escenarios

Los siguientes tres [conceptos](#) básicos subyacen a la funcionalidad de transmisión en tiempo real: escenario, estrategia y renderizador. El objetivo del diseño es minimizar la cantidad de lógica necesaria por parte del cliente para crear un producto que funcione.

Stage.Strategy

La implementación de `Stage.Strategy` es sencilla:

```

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  
}
```

En resumen, la publicación depende del estado de la variable `publishEnabled` interna y, si publicamos, publicaremos las transmisiones que recopilamos anteriormente. Por último, en este ejemplo, siempre nos suscribimos a otros participantes y recibimos tanto su contenido de audio como de video.

StageRenderer

La implementación de `StageRenderer` también es bastante simple, aunque, dada la cantidad de funciones, contiene bastante más código. El enfoque general de este renderizador es actualizar `ParticipantAdapter` cuando el SDK nos notifique un cambio en un participante. Hay ciertos casos en los que tratamos a los participantes locales de forma diferente, porque hemos decidido administrarlos por nuestra cuenta para que puedan ver la vista previa de la cámara antes de unirse.

```
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) {}
}
```

Implementación de layoutManager de RecyclerView personalizado

Establecer diferentes números de participantes puede resultar complejo. Lo ideal es que ocupen todo el marco de la vista principal, pero gestionar la configuración de cada participante de forma independiente no es la mejor forma de lograrlo. Para facilitarlo, veremos cómo implementar `RecyclerView.LayoutManager`.

Cree otra clase, `StageLayoutManager`, que debería ampliar `GridLayoutManager`. Esta clase está pensada para calcular cómo se mostrará cada participante en función del número de participantes en un diseño de filas o columnas basado en flujos. Cada fila tiene la misma altura que las demás, pero las columnas pueden tener diferentes anchuras por fila. Consulte el comentario del código que aparece sobre la variable `layouts` para ver cómo personalizar este comportamiento.

```
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 curPosition = position
```

```

        while (curPosition - config[row] >= 0) {
            curPosition -= 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
}

```

En `MainActivity.kt`, tenemos que configurar el adaptador y el administrador de diseño de `RecyclerView`:


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

Enlace de acciones de la interfaz de usuario

Ya casi está, solo tenemos que enlazar algunas acciones de la interfaz de usuario.

Primero, tenemos que hacer que MainActivity supervise los cambios en StateFlow desde MainViewModel:

```
// 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)
        }
    }
}
```

A continuación, agregamos los oyentes a nuestro botón de unión y a la casilla de publicación:

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

Ambos elementos llaman a funciones en MainViewModel, lo cual implementaremos ahora:

```
internal fun joinStage(token: String) {
    if (_connectionState.value != Stage.ConnectionState.DISCONNECTED) {
        // If we're already connected to a stage, leave it.
        stage?.leave()
    } else {
        if (token.isEmpty()) {
            Toast.makeText(getApplication(), "Empty Token", Toast.LENGTH_SHORT).show()
            return
        }
    }
}
```

```

    }
    try {
        // Destroy the old stage first before creating a new one.
        stage?.release()
        val stage = Stage(getApplication(), token, this)
        stage.addRenderer(this)
        stage.join()
        this.stage = stage
    } catch (e: BroadcastException) {
        Toast.makeText(getApplication(), "Failed to join stage
${e.localizedMessage}", Toast.LENGTH_LONG).show()
        e.printStackTrace()
    }
}

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

```

Renderización de los participantes

Por último, tenemos que renderizar los datos que recibimos del SDK en el elemento de participante que creamos anteriormente. Ya hemos completado la lógica de RecyclerView, por lo que solo tenemos que implementar la API bind en ParticipantItem.

Empezamos por agregar la función vacía y, luego, la analizaremos paso a paso:

```

fun bind(participant: StageParticipant) {
}

```

Primero, analizaremos el estado sencillo, el ID del participante, el estado de la publicación y el estado de la suscripción. Para hacerlo, tan solo actualizamos TextViews directamente:

```

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
```

A continuación, actualizaremos los estados silenciados de audio y video. Para obtener el estado silenciado, tenemos que encontrar el `ImageDevice` y el `AudioDevice` de la matriz de transmisiones. Para optimizar el rendimiento, recordamos los últimos ID de los dispositivos conectados.

```
// 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"
}
```

Por último, queremos renderizar una vista previa de `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
```

Y mostramos las estadísticas de audio de `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
```

Publicación y suscripción con el SDK de transmisión para iOS de IVS

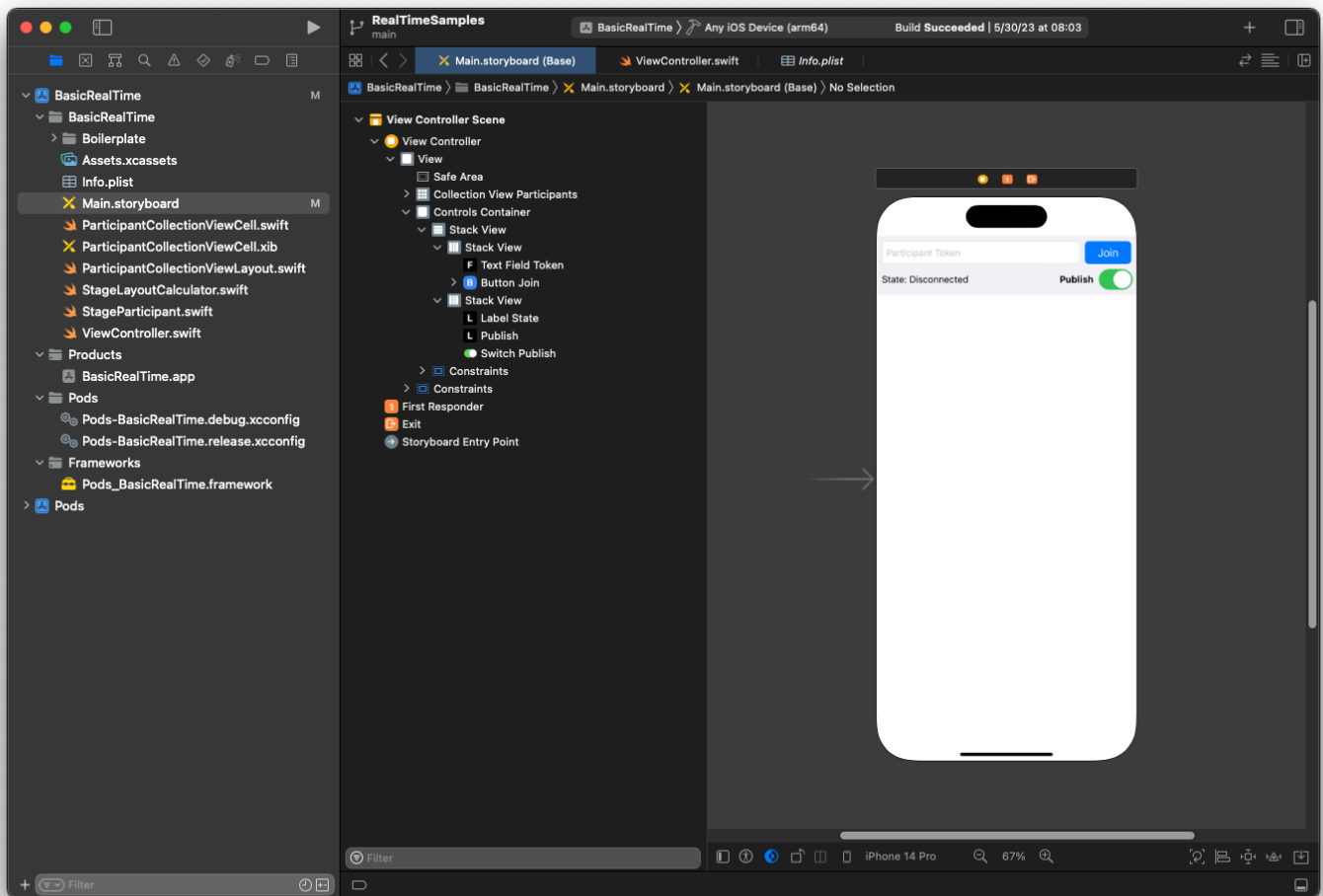
En esta sección se explican los pasos necesarios para publicar y suscribirse a una fase mediante una aplicación iOS.

Creación de vistas

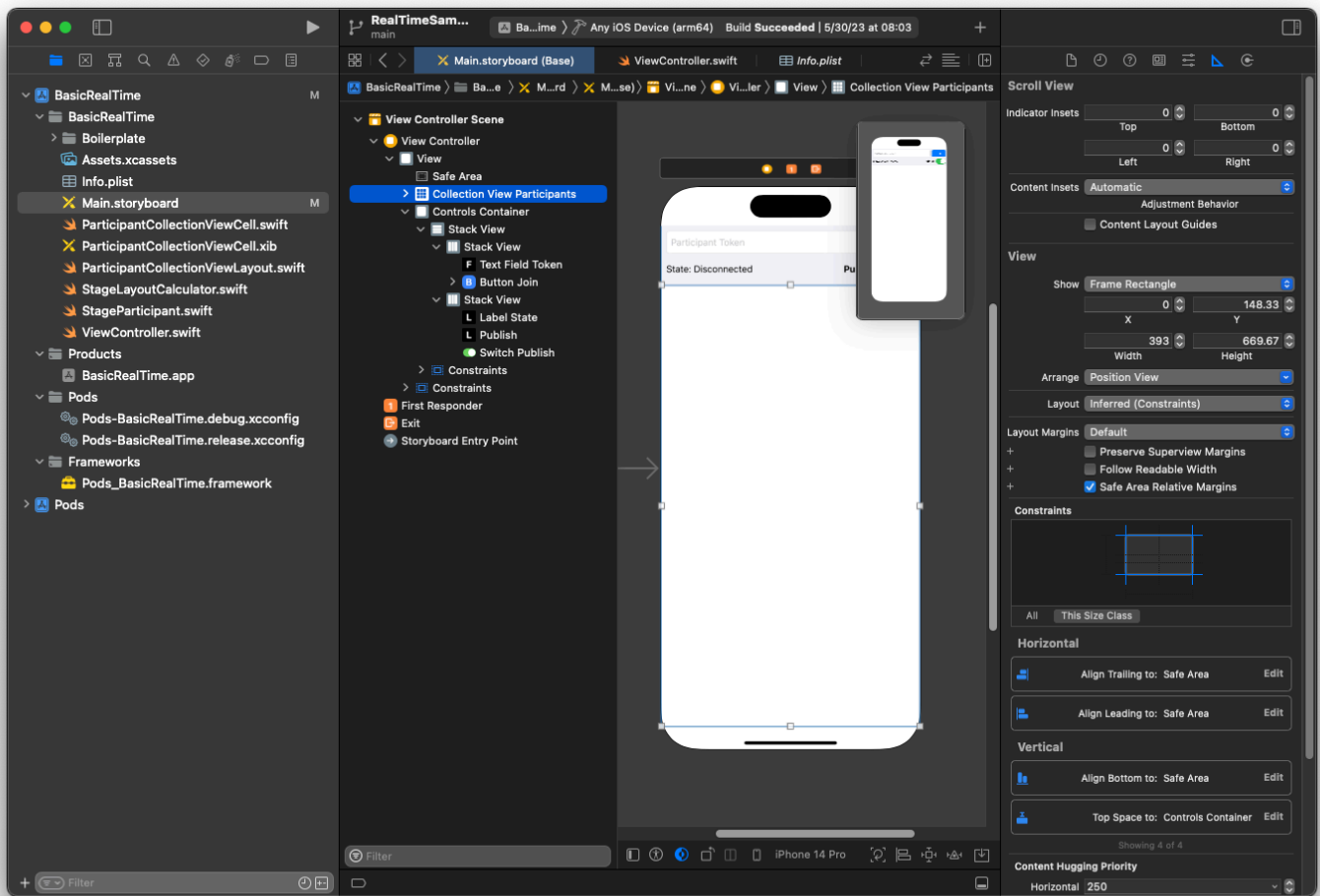
Lo primero es usar el archivo `ViewController.swift` creado automáticamente para importar `AmazonIVSBroadcast` y, luego, agregar algunas `@IBOutlet`s que vincular:

```
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!
```

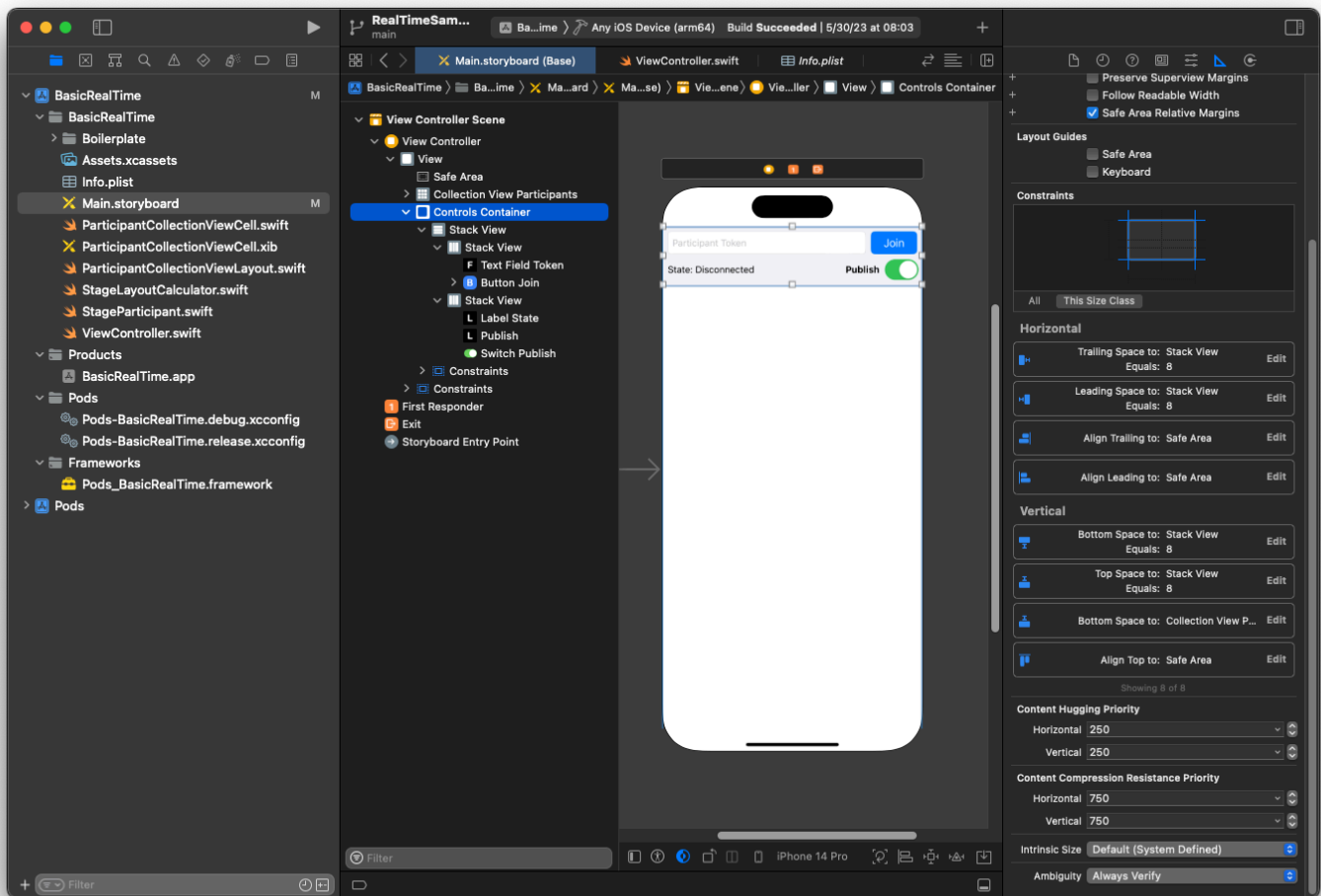
Ahora creamos esas vistas y las vinculamos en `Main.storyboard`. Esta es la estructura de vistas que utilizaremos:



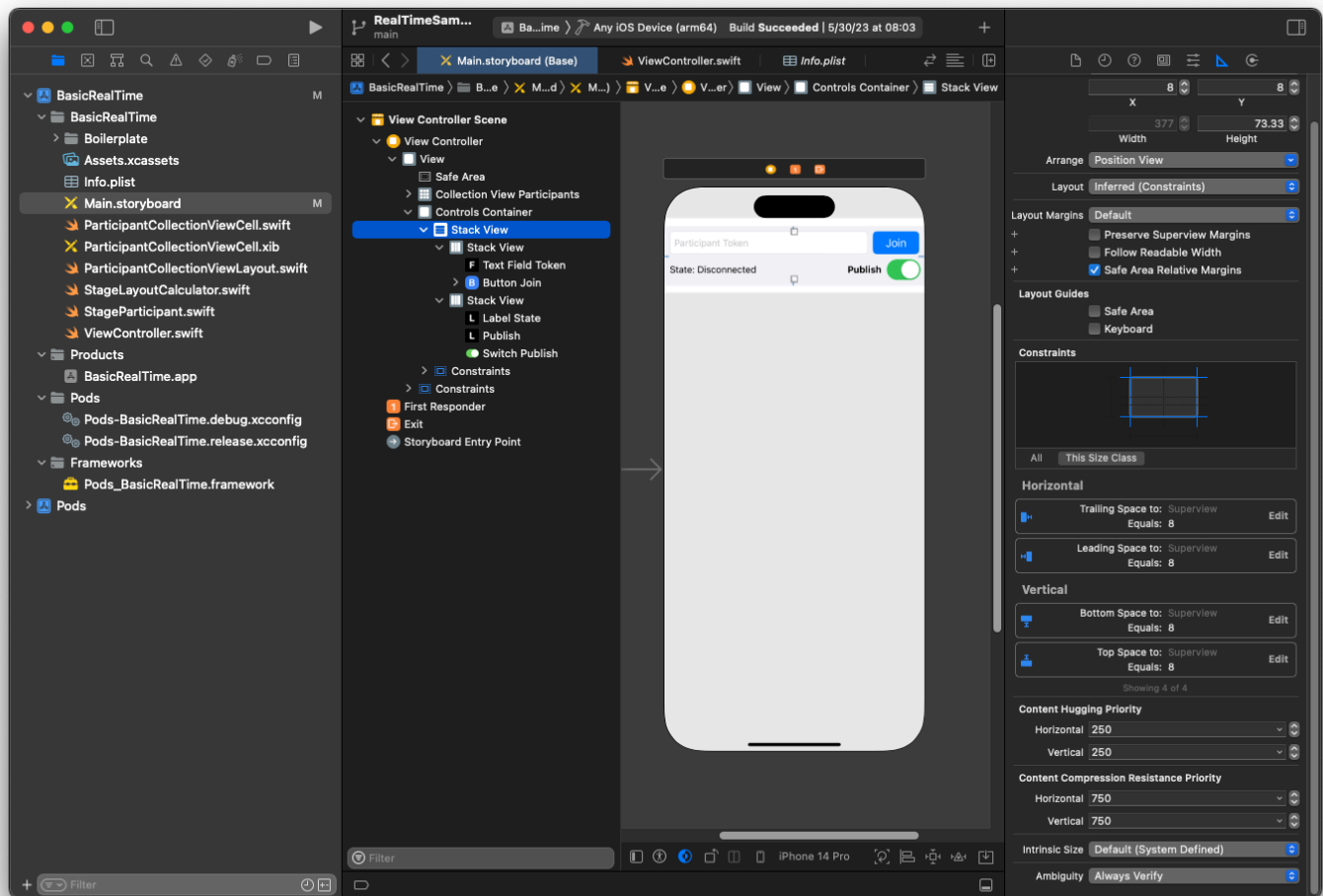
Para configurar AutoLayout, tenemos que personalizar tres vistas. La primera vista es Vista de colección de participantes (una UICollectionView). Alinee Inicial, Final e Inferior con respecto a Área segura. Alinee también Superior con respecto a Contenedor de controles.



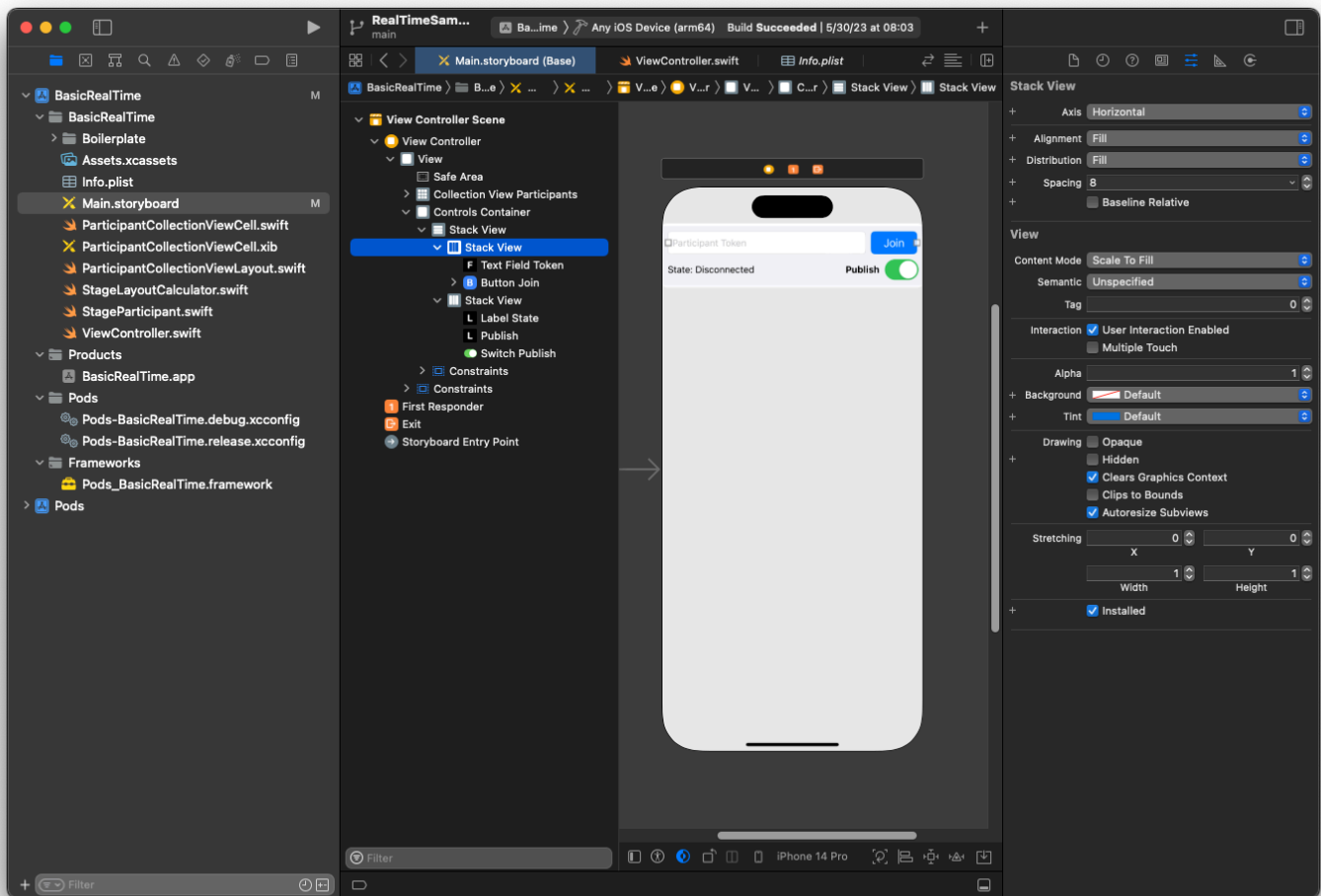
La segunda vista es Contenedor de controles. Alinee Inicial, Final y Superior con respecto a Área segura:



La tercera y última vista es Vista de pila vertical. Alinee Superior, Inicial, Final e Inferior con respecto a Supervista. En cuanto al diseño, establezca el espaciado en 8 en lugar de en 0.



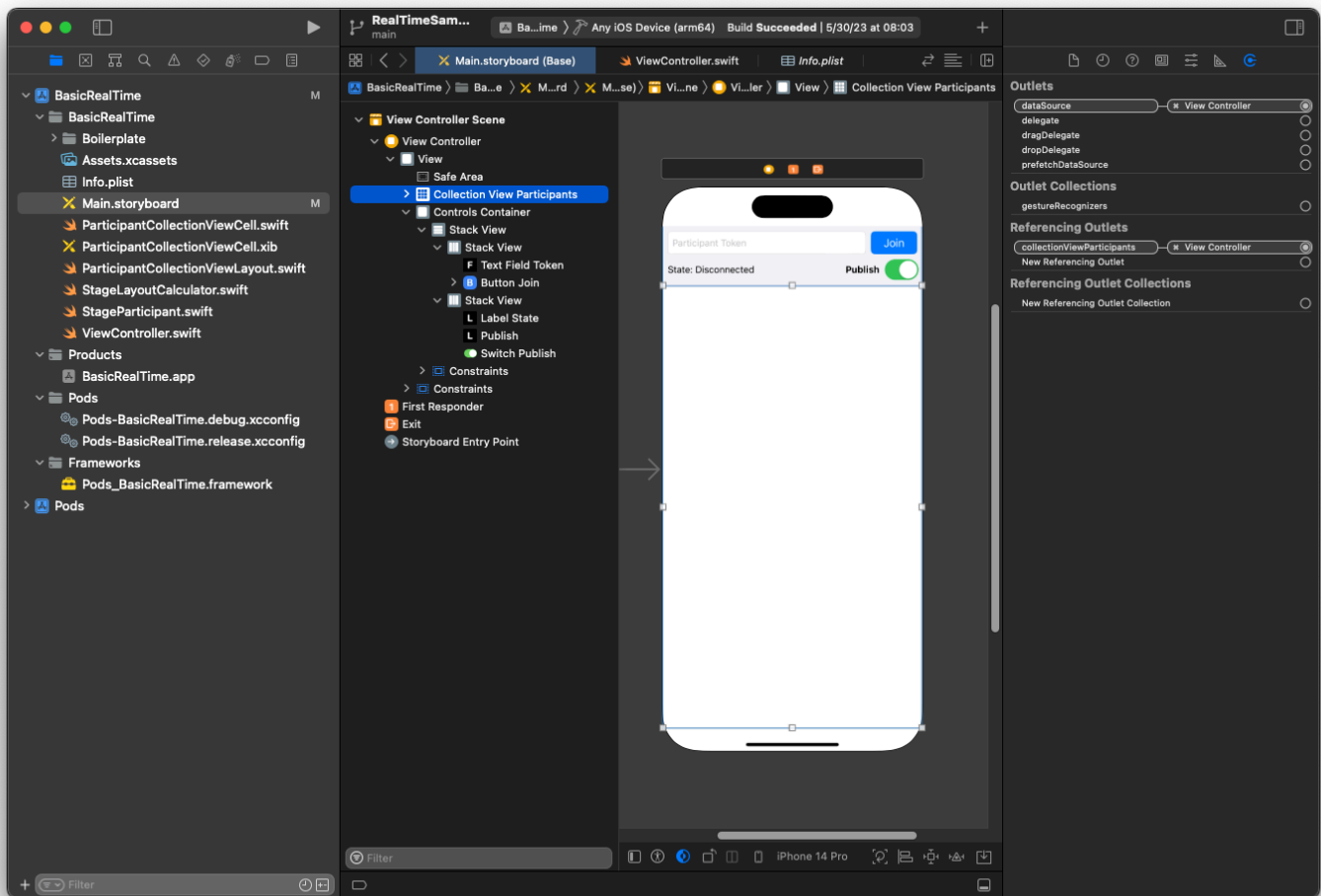
Las UIStackViews se encargarán del diseño de las vistas restantes. En las tres UIStackViews, seleccione la opción Relleno para Alineación y Distribución.



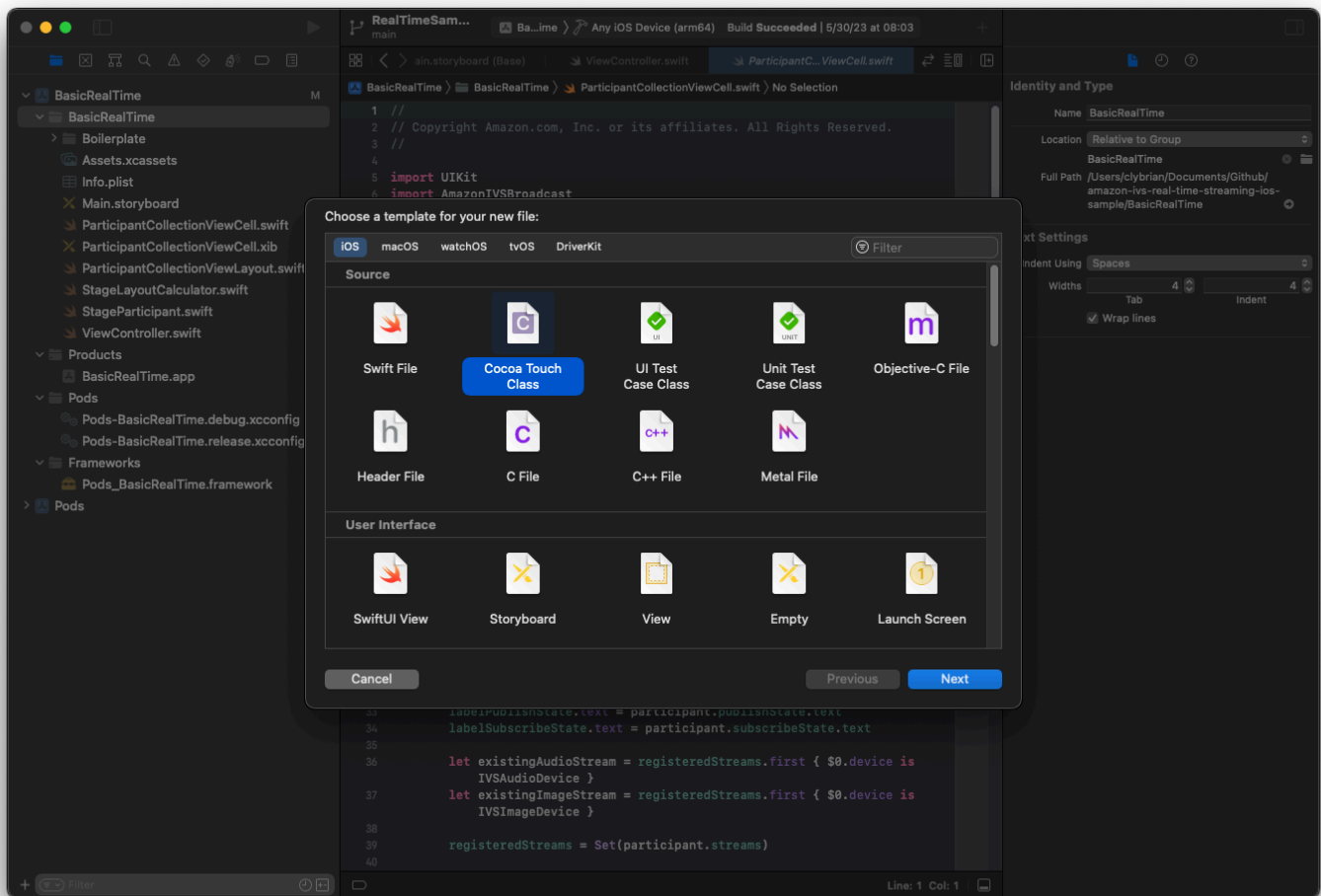
Por último, vinculemos estas vistas a nuestro `ViewController`. Desde arriba, asigne las siguientes vistas:

- El campo de texto de unión se asigna a `textFieldToken`.
- El botón de unión se asigna a `buttonJoin`.
- La etiqueta de estado se asigna a `labelState`.
- El botón de publicación se asigna a `switchPublish`.
- Vista de colección de participantes se asigna a `collectionViewParticipants`.

Aproveche también para establecer el `dataSource` del elemento Vista de colección de participantes en el `ViewController` propietario:



Ahora creamos la subclase `UICollectionViewCell` en la que se renderizarán los participantes. Lo primero es crear un nuevo archivo Cocoa Touch Class:



Asígnele el nombre `ParticipantUICollectionViewCell` y conviértala en una subclase de `UICollectionViewCell` en Swift. Volvemos al archivo de Swift y creamos `@IBOutlet`s que vincular:

```

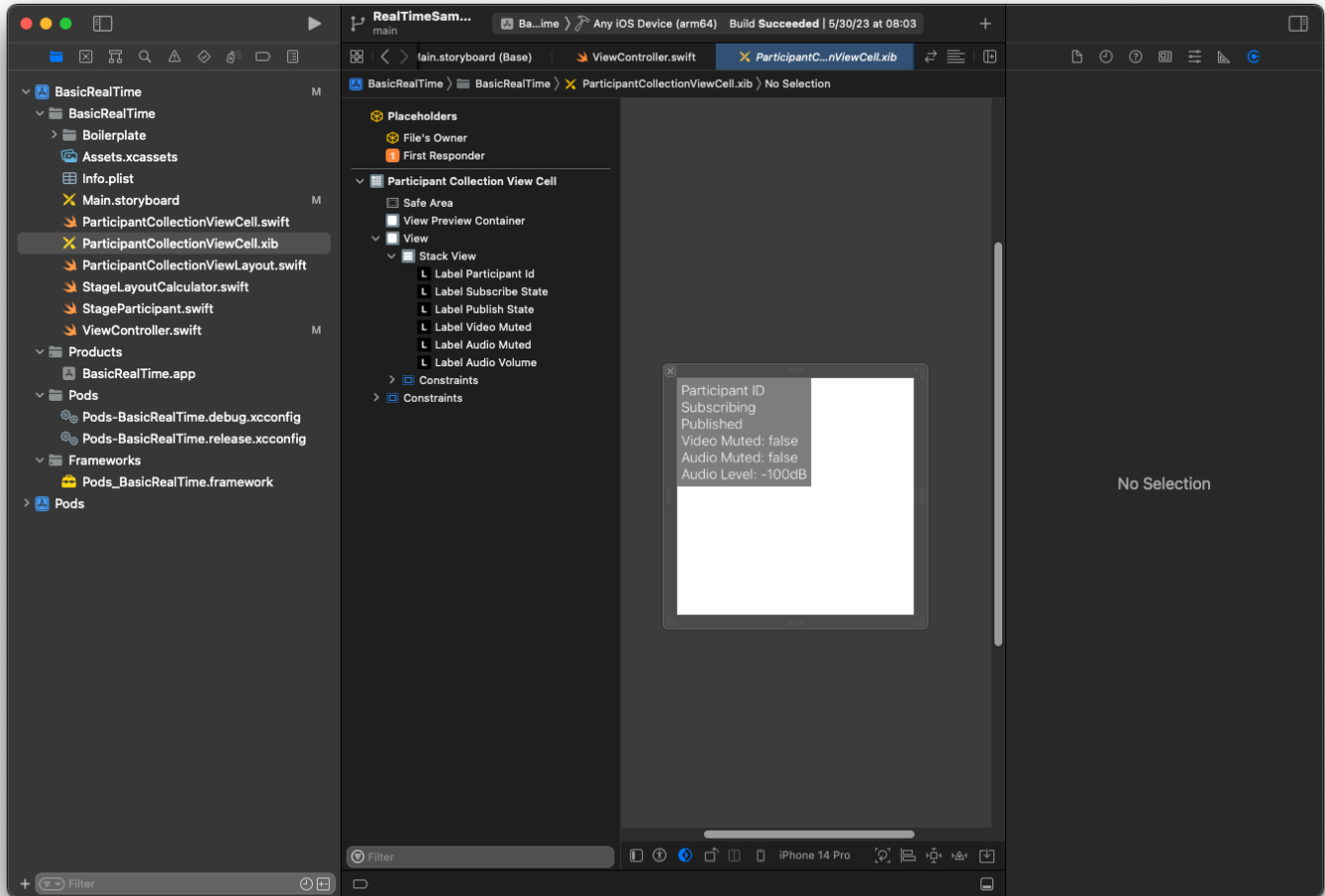
import AmazonIVSBroadcast

class ParticipantCollectionViewCell: UICollectionViewCell {

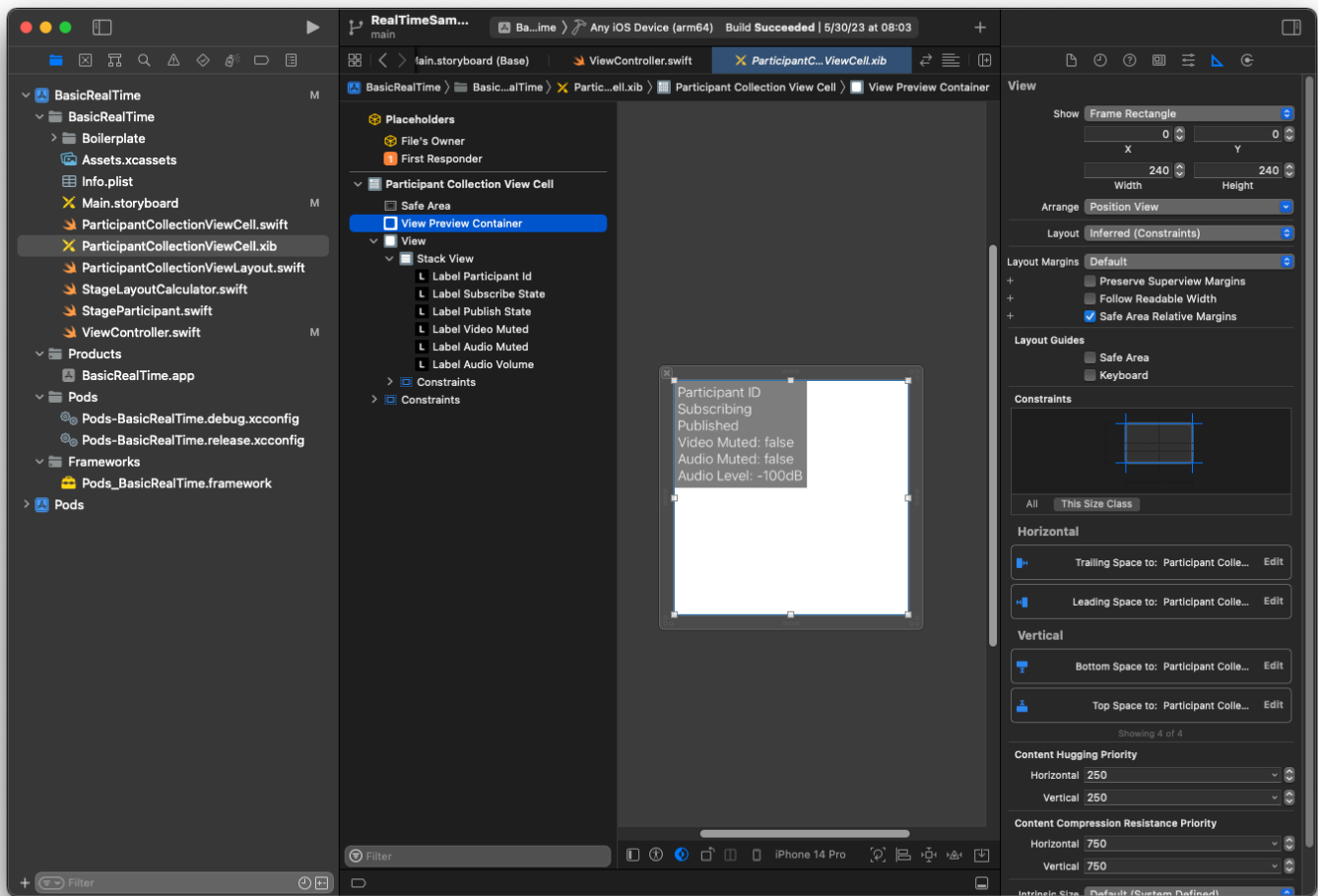
    @IBOutlet private var viewPreviewContainer: UIView!
    @IBOutlet private var labelParticipantId: UILabel!
    @IBOutlet private var labelSubscribeState: UILabel!
    @IBOutlet private var labelPublishState: UILabel!
    @IBOutlet private var labelVideoMuted: UILabel!
    @IBOutlet private var labelAudioMuted: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

```

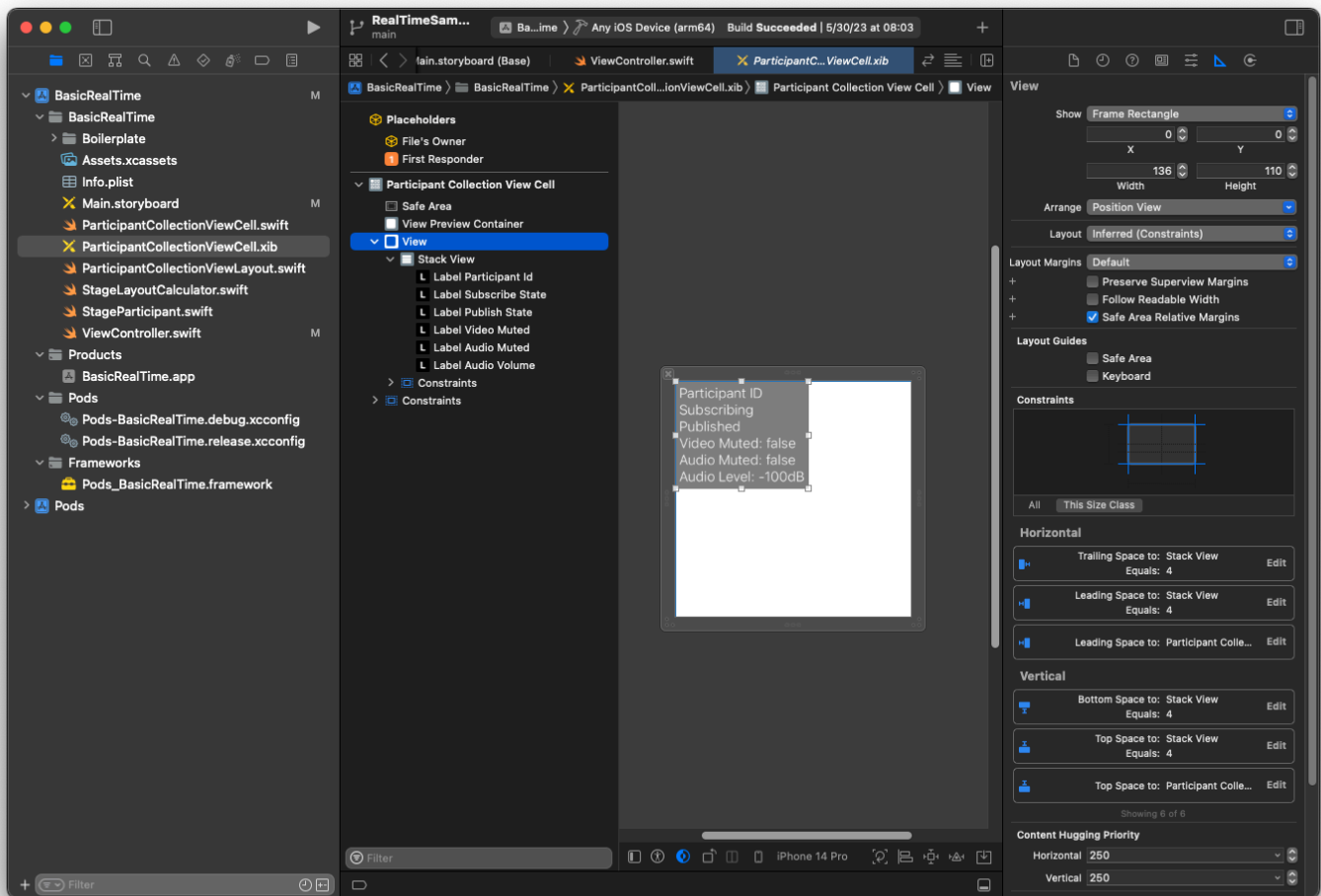
En el archivo XIB asociado, cree esta jerarquía de vistas:



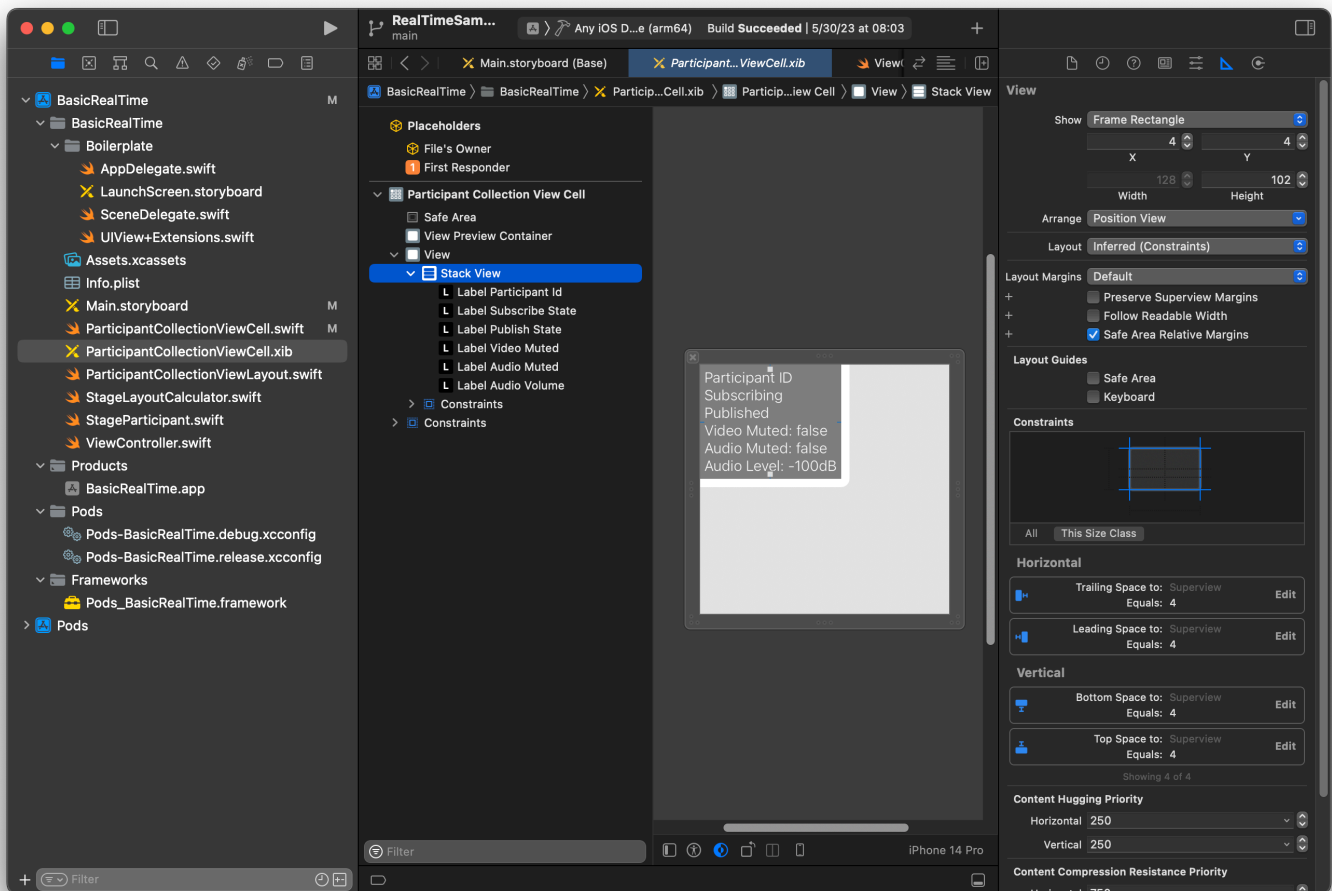
En AutoLayout, volveremos a modificar tres vistas. La primera vista es Vista previa de contenedores. Alinee Final, Inicial, Superior e Inferior con respecto a Celda de vista de colección de participantes.



La segunda vista es Vista. Alinee Inicial y Superior con respecto a Celda de vista de colección de participantes y cambie el valor a 4.



La tercera vista es Vista de pila. Alinee Final, Inicial, Superior e Inferior con respecto a Supervista y cambie el valor a 4.



Permisos y temporizador de inactividad

En `ViewController`, tenemos que desactivar el temporizador de inactividad del sistema para evitar que el dispositivo entre en estado de suspensión mientras se esté utilizando nuestra aplicación:

```
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
}
```

A continuación, solicitamos los permisos de cámara y micrófono del sistema:

```

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)
    }
}

```

Estado de la aplicación

Tenemos que configurar `collectionViewParticipants` con el archivo de diseño que creamos anteriormente:

```

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")
}
```

Para representar a cada participante, creamos una estructura simple llamada `StageParticipant`. Esta puede incluirse en el archivo `ViewController.swift`, aunque también se puede crear un archivo nuevo.

```
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
    }
}
```

Para hacer un seguimiento de esos participantes, los guardamos en una matriz que sea una propiedad privada de `ViewController`:

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

Esta propiedad se utilizará para facilitar información al `UICollectionViewDataSource` que enlazamos en el guion gráfico anterior:

```
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'")
        }
    }
}
}

```

Para ver su propia vista previa antes de unirse a un escenario, creamos inmediatamente un participante local:

```

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

```

Esto hace que la celda de un participante se renderice inmediatamente cuando se ejecute la aplicación. La celda representa al participante local.

Los usuarios quieren poder verse antes de unirse a un escenario, por lo que, a continuación, implementamos el método `setupLocalUser()`. El código de gestión de permisos anterior se encarga de llamar a este método. Almacenamos la referencia de la cámara y el micrófono como objetos de `IVSLocalStageStream`.

```

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)
}

```

Con esto detectamos la cámara y el micrófono del dispositivo a través del SDK y los almacenamos en nuestro objeto `streams` local. A continuación, asignamos la matriz `streams` del primer participante (el participante local que creamos anteriormente) a `streams`. Por último, llamamos a `participantsChanged` con un `index` de 0 y un `changeType` con el valor `updated`. Esta es una función auxiliar para actualizar `UICollectionView` con bonitas animaciones. Este es el resultado:

```

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)])
    }
}

```

No se preocupe por `cell.set` aún, lo veremos luego, pero ahí es donde renderizaremos el contenido de la celda en función del participante.

`ChangeType` es una enumeración simple:

```

enum ChangeType {

```

```

    case joined, updated, left
  }

```

Lo último es ver si el escenario está conectado. Usamos un simple valor `bool` para comprobarlo. Cuando este valor se actualice, actualizará automáticamente nuestra interfaz de usuario.

```

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

```

Implementación del SDK de escenarios

Los siguientes tres [conceptos](#) básicos subyacen a la funcionalidad de transmisión en tiempo real: escenario, estrategia y renderizador. El objetivo del diseño es minimizar la cantidad de lógica necesaria por parte del cliente para crear un producto que funcione.

IVSStageStrategy

La implementación de `IVSStageStrategy` es sencilla:

```

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
  }
}

```

```

    }
}

```

En resumen, solo publicamos si la opción de publicación está activada. Si publicamos, publicaremos las transmisiones que hemos recopilado anteriormente. Por último, en este ejemplo, siempre nos suscribimos a otros participantes y recibimos tanto su contenido de audio como de video.

IVSStageRenderer

La implementación de `IVSStageRenderer` también es bastante simple, aunque, dada la cantidad de funciones, contiene bastante más código. El enfoque general de este renderizador es actualizar la matriz `participants` cuando el SDK nos notifique un cambio en un participante. Hay ciertos casos en los que tratamos a los participantes locales de forma diferente, porque hemos decidido administrarlos por nuestra cuenta para que puedan ver la vista previa de la cámara antes de unirse.

```

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)
```

```
    }  
}
```

Este código usa una extensión para convertir el estado de conexión en texto legible para nosotros:

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

Implementación de UICollectionViewLayout personalizado

Establecer diferentes números de participantes puede resultar complejo. Lo ideal es que ocupen todo el marco de la vista principal, pero gestionar la configuración de cada participante de forma independiente no es la mejor forma de lograrlo. Para facilitarlo, veremos cómo implementar UICollectionViewLayout.

Cree otro archivo, ParticipantCollectionViewLayout.swift, que debería ampliar UICollectionViewLayout. Esta clase usará otra clase llamada StageLayoutCalculator, que veremos pronto. La clase recibe los valores de marco calculados para cada participante y, a continuación, genera los objetos de UICollectionViewLayoutAttributes necesarios.

```
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))
            }
        }
    }
}

```

La clase `StageLayoutCalculator.swift` es más importante. Esto está pensado para calcular los marcos de cada participante en función del número de participantes en un diseño de filas o columnas basado en flujos. Cada fila tiene la misma altura que las demás, pero las columnas

pueden tener diferentes anchuras por fila. Consulte el comentario del código que aparece sobre la variable `layouts` para ver cómo personalizar este comportamiento.

```
import Foundation
import UIKit

class StageLayoutCalculator {

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

```

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

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

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

    let halfPadding = padding / 2.0

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

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

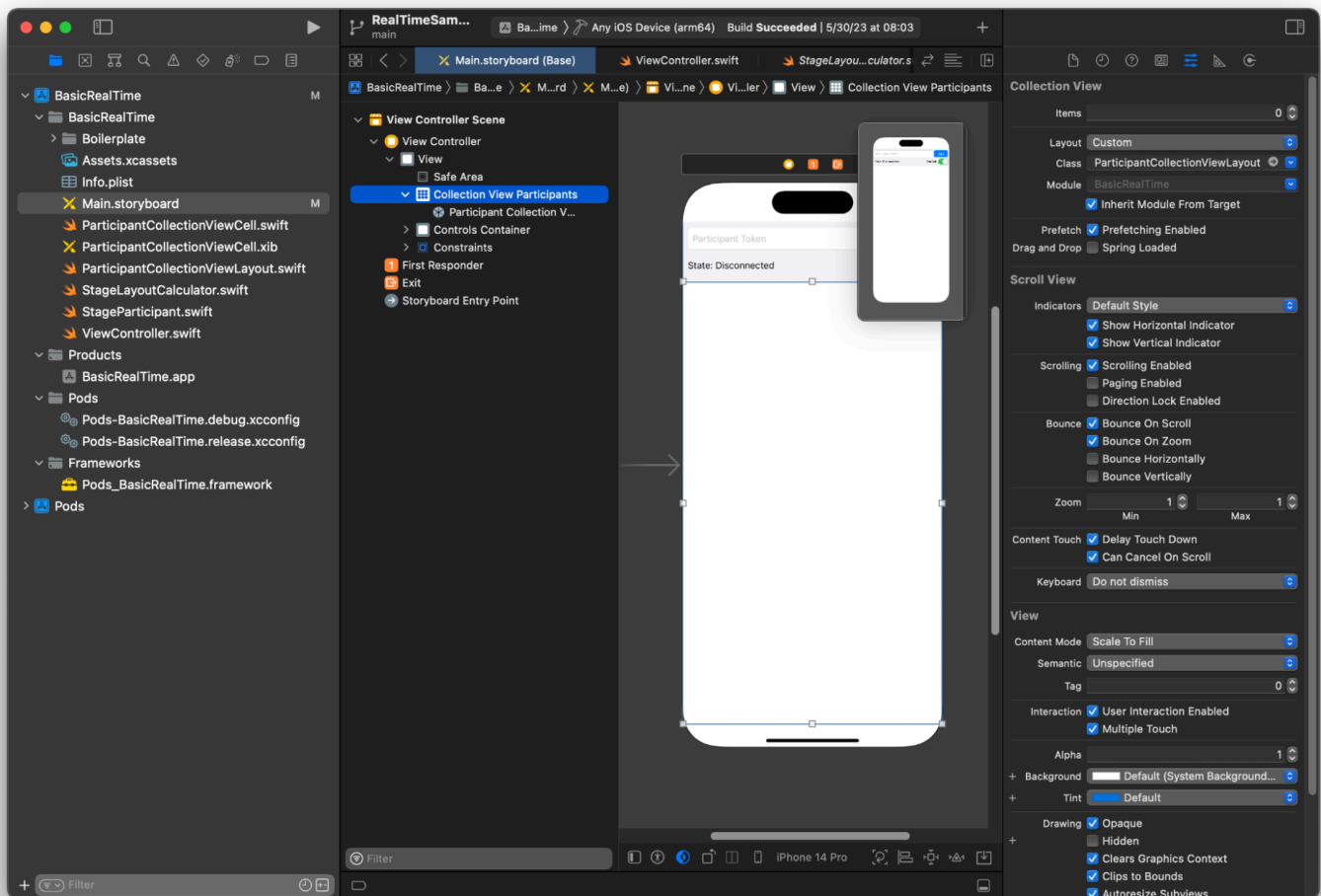
```

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

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

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

En `Main.storyboard`, asegúrese de establecer la clase que acabamos de crear como la clase de diseño de `UICollectionView`:



Enlace de acciones de la interfaz de usuario

Ya casi hemos terminado, pero aún tenemos que crear unas cuantas IBActions.

Primero, nos encargamos del botón de unión. Responde de manera diferente en función del valor de `connectingOrConnected`. Cuando está conectado, simplemente abandona el escenario. Si está desconectado, lee el texto del token `UITextField` y crea un nuevo `IVSStage` con ese texto. A continuación, agregamos `ViewController` como `strategy`, `errorDelegate` y renderizador para `IVSStage`. Finalmente, nos unimos al escenario de forma asíncrona.

```
@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)")
    }
}
}

```

La otra acción de la interfaz de usuario que tenemos que enlazar es la opción de publicación:

```

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

```

Renderización de los participantes

Por último, tenemos que renderizar los datos que recibimos del SDK en la celda de participante que creamos anteriormente. Ya hemos completado la lógica de `UICollectionView`, por lo que solo tenemos que implementar la API `set` en `ParticipantCollectionViewCell.swift`.

Empezamos por agregar la función `empty` y, luego, la analizaremos paso a paso:

```

func set(participant: StageParticipant) {
}

```

Primero, analizamos el estado sencillo, el ID del participante, el estado de la publicación y el estado de la suscripción. Para hacerlo, tan solo actualizamos `UILabels` directamente:

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

Las propiedades de texto de las enumeraciones de publicación y suscripción provienen de extensiones locales:

```
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()
        }
    }
}
```

A continuación, actualizamos los estados silenciados de audio y video. Para obtener los estados silenciados, tenemos que encontrar el `IVSImageDevice` y el `IVSAudioDevice` de la matriz `streams`. Para optimizar el rendimiento, recordaremos los últimos dispositivos conectados.

```
// 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)"
```

Por último, queremos renderizar una vista previa de `imageDevice` y mostrar las estadísticas de audio de `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"
}
```

La última función que tenemos que crear es `updatePreview()`, que agrega una vista previa del participante a nuestra vista:

```
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)
        }
    }
}
```

```
    }  
}
```

Lo anterior usa una función auxiliar en UIView para facilitar la incorporación de subvistas:

```
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),  
        ])  
    }  
}
```

Supervisión del streaming en tiempo real de Amazon IVS

Este documento proporciona detalles sobre las opciones disponibles para monitorear su aplicación de transmisión en tiempo real de IVS.

¿Qué es la sesión de una fase?

La sesión de una fase comienza cuando el primer participante se une a una fase y finaliza unos minutos después de que el último participante deje de publicar en dicha fase. Las sesiones de las fases ayudan a depurar las fases de larga duración al separar los eventos y los participantes en sesiones de corta duración.

Ver sesiones de fases y participantes

Instrucciones de la consola

1. Abra la [consola de Amazon IVS](#).

(También puede acceder a la consola de Amazon IVS a través de la [Consola de administración de AWS](#)).

2. En el panel de navegación elija Fases. (Si el panel de navegación está contraído, ábralo primero eligiendo el icono de hamburguesa).
3. Elija la fase para ir a la página de detalles correspondiente.
4. Desplácese hacia abajo en la página hasta que vea la sección Sesiones de fase y, a continuación, seleccione la sesión de una fase para ver la página de detalles.
5. Para ver los participantes de la sesión, desplácese hacia abajo hasta que vea la sección Participantes y, a continuación, seleccione un participante para ver la página de detalles pertinente, incluidos los gráficos de las métricas de Amazon CloudWatch.

Ver eventos de un participante

Los eventos se envían cuando el estado de un participante en de un escenario cambia. Por ejemplo, cuando se une a un escenario o se produce un error al intentar publicar en uno de ellos. No todos los errores provocan eventos; por ejemplo, los errores de red del lado del cliente y los errores de firma

de los tokens no se envían como eventos. Para gestionar estos errores en la aplicación cliente, utilice los [SDK de transmisión de IVS](#).

Instrucciones de la consola

1. Vaya a la página de detalles del participante tal y como se ha indicado anteriormente.
2. Desplácese hacia abajo hasta que vea la sección Eventos. Se muestra una lista ordenada de los eventos del participante. Consulte [Uso de Amazon EventBridge con Amazon IVS](#) para obtener más información sobre los eventos que se emiten para los participantes.

Instrucciones de la CLI

El acceso a los eventos de sesiones de escenarios con la AWS CLI es una opción avanzada y requiere que antes descargue y configure la CLI en su equipo. Para obtener más información, consulte la [Guía del usuario de la Interfaz de la línea de comandos de AWS](#).

1. Enumere las sesiones de una fase para encontrar una en particular:

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

2. Enumere los participantes de la sesión de una fase para encontrar un participante:

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

3. Enumere los eventos de la sesión de una fase y un participante:

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

A continuación, se muestra una respuesta de ejemplo a la llamada `list-participant-events`:

```
{
  "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"
  }
]
```

Acceso a métricas de CloudWatch

Para que las métricas de CloudWatch estén disponibles, se requieren las siguientes versiones del SDK de transmisión de IVS: Web 1.5.0 o posterior, Android 1.12.0 o posterior o iOS 1.12.0 o posterior.

Instrucciones de la consola de CloudWatch

1. Abra la consola de CloudWatch en <https://console.aws.amazon.com/cloudwatch/>.
2. En el panel de navegación lateral, expanda el menú desplegable Metrics (Métricas) y, a continuación, seleccione All metrics (Todas las métricas).
3. En la pestaña Explorar, mediante el menú desplegable sin etiqueta de la izquierda, seleccione su región de "inicio", donde se crearon los canales. Para obtener más información sobre las regiones, consulte [Solución global, control regional](#). Para obtener una lista de las regiones admitidas, consulte la [Página de Amazon IVS](#) en la Referencia general de AWS.
4. En la parte inferior de la pestaña Explorar, seleccione el espacio de nombres IVSRealTime.
5. Realice una de las siguientes acciones siguientes:
 - a. En la barra de búsqueda, ingrese el ID de recurso (parte del ARN, `arn:::ivs:stage/<resource id>`).

A continuación, seleccione `IVSRealTime > Estado de las métricas`.

- b. Si `IVSRealTime` aparece como un servicio seleccionable en Espacios de nombres de AWS, selecciónelo. Se mostrará si utiliza transmisión en tiempo real de Amazon IVS y envía métricas a Amazon CloudWatch. (Si `IVSRealTime` no está en la lista, no tiene ninguna métrica de Amazon IVS).

Luego, elija la agrupación de dimensiones que desee; las dimensiones disponibles se muestran a continuación en [Métricas de CloudWatch](#).

6. Elija métricas para agregarlas al gráfico. Las métricas disponibles se muestran a continuación en [Métricas de CloudWatch](#).

También puede acceder al gráfico de CloudWatch de la sesión de transmisión desde la página de detalles de la sesión de transmisión, seleccionando el botón `View in CloudWatch` (Ver en CloudWatch).

Instrucciones de la CLI

También puede obtener acceso a las métricas mediante la AWS CLI. Esto requiere que primero descargue y configure la CLI en su equipo. Para obtener más información, consulte la [Guía del usuario de la interfaz de línea de comandos de AWS](#).

A continuación, para obtener acceso a las métricas de transmisión en tiempo real de Amazon IVS mediante AWS CLI:

- En el símbolo del sistema, ejecute:

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

Para obtener más información, consulte [Uso de las métricas de Amazon CloudWatch](#) en la Guía del usuario de Amazon CloudWatch.

Métricas de CloudWatch: streaming en tiempo real de IVS

Amazon IVS proporciona las siguientes métricas en el espacio de nombres de `AWS/IVSRealTime`.

Para que las métricas de CloudWatch estén disponibles, se debe usar SDK de transmisión web 1.5.2 o una versión posterior.

La dimensión puede tener los siguientes valores válidos:

- La dimensión Stage es un ID de recurso (parte del ARN, `arn:::stage/<resource id>`).
- La dimensión de Participant es una participantID.
- SimulcastLayer es “alta”, “media”, “baja” o “sin RID” para el MediaType “video” o “deshabilitada” para el MediaType “audio” Este valor también puede estar vacío.
- La dimensión de MediaType es “video” o “audio” (cadena).

Métrica	Dimensión	Descripción
DownloadPacketLoss	Stage	<p>Cada muestra representa el porcentaje de paquetes que perdió un suscriptor determinado durante la descarga del servidor de IVS.</p> <p>Unidad: porcentaje</p> <p>Estadísticas válidas: promedio, máximo, mínimo: número medio, número mayor o menor (respectivamente) pérdida de paquetes durante el intervalo configurado</p>
DownloadPacketLoss	Stage, Participant	<p>Filtra DownloadPacketLoss por participante, para los suscriptores que también son publicadores. Las muestras representan el porcentaje de paquetes que perdió un suscriptor durante la descarga del servidor de IVS. Las muestras se emiten solo cuando el participante también es publicador.</p> <p>Unidad: porcentaje</p> <p>Estadísticas válidas: promedio, máximo, mínimo: número medio, número mayor o menor (respectivamente) de fotogramas descartados durante el intervalo configurado</p>
DroppedFrames	Stage	<p>Cada muestra representa el porcentaje de fotogramas descartados por un suscriptor determinado.</p> <p>Unidad: porcentaje</p>

Métrica	Dimensión	Descripción
		Estadísticas válidas: promedio, máximo, mínimo: número medio, número mayor o menor (respectivamente) de fotogramas descartados durante el intervalo configurado
DroppedFrames	Stage, Participant	<p>Filtra DroppedFrames por participante, para los suscriptores que también son publicadores. Los ejemplos representan el porcentaje de fotogramas que se eliminaron entre el participante suscriptor y todos los editores del escenario. Las muestras se emiten solo cuando el participante también es publicador.</p> <p>Unidad: porcentaje</p> <p>Estadísticas válidas: promedio, máximo, mínimo: número medio, número mayor o menor (respectivamente) de fotogramas descartados durante el intervalo configurado</p>
PublishBps	Stage	<p>Las muestras emitidas representan la velocidad total a la que un publicador envía datos de video y audio (sumados en todas las capas de transmisión simultánea).</p> <p>Unidad: bits por segundo</p> <p>Estadísticas válidas: promedio, máximo, mínimo: número medio, número mayor o menor (respectivamente) de tasas de bits durante el intervalo configurado</p>

Métrica	Dimensión	Descripción
PublishBitrate	Stage, Participant, Simulcast Layer, MediaType	<p>Filtra PublishBitrate por participante, capa de transmisión simultánea y tipo de medio. El ID de la capa de transmisión simultánea lo establece el SDK de transmisión. Cuando la transmisión simultánea está deshabilitada, este ID de capa se establece como “deshabilitado”. El tipo de medio es video o audio.</p> <p>Unidad: bits por segundo</p> <p>Estadísticas válidas: promedio, máximo, mínimo: número medio, número mayor o menor (respectivamente) de tasas de bits durante el intervalo configurado</p>
PublishFrameRate	Stage, Participant	<p>Frecuencia con la que se reciben fotogramas de video de un publicador determinado. Esta métrica solo está disponible para los participantes que publican en RTMP.</p> <p>Unidad: recuento por segundo</p> <p>Estadísticas válidas: promedio, máximo, mínimo: número promedio, número mayor o número menor (respectivamente) de velocidad de fotogramas durante el intervalo configurado</p>
Publishers	Stage	<p>Número de participantes que publican en el escenario.</p> <p>Unidad: recuento</p> <p>Estadísticas válidas: Promedio, Máximo, Mínimo</p>

Métrica	Dimensión	Descripción
PublishResolution	Stage, Participant, Simulcast Layer, MediaType	<p>Número de píxeles en el ancho o alto más pequeño del marco. Por ejemplo, para un marco horizontal de 1920 x 1080, la PublishResolution es 1080. Para un marco de retrato de tamaño 720 x 1280, la PublishResolution es 720.</p> <p>Unidad: recuento</p> <p>Estadísticas válidas: Promedio, Máximo, Mínimo</p>
SubscribeBitrate	Stage	<p>Las muestras emitidas representan la velocidad total a la que un suscriptor determinado recibe datos de video y audio.</p> <p>Unidad: bits por segundo</p> <p>Estadísticas válidas: promedio, máximo, mínimo: número medio, número mayor o menor (respectivamente) de tasas de bits durante el intervalo configurado</p>
SubscribeBitrate	Stage, Participant, MediaType	<p>Filtra <code>SubscribeBitrate</code> por participante, para los suscriptores que también son publicadores. Las muestras representan la velocidad de bits a la que un suscriptor determinado recibe el <code>MediaType</code> determinado. Las muestras solo se emiten mientras el participante suscriptor también es publicador.</p> <p>Unidad: bits por segundo</p> <p>Estadísticas válidas: promedio, máximo, mínimo: número medio, número mayor o menor (respectivamente) de tasas de bits durante el intervalo configurado</p>

Métrica	Dimensión	Descripción
Subscribers	Stage	<p>Número de participantes suscritos al escenario. Tenga en cuenta que los participantes que publican y se suscriben activamente se cuentan tanto como editores como suscriptores.</p> <p>Unidad: recuento</p> <p>Estadísticas válidas: Promedio, Máximo, Mínimo</p>

SDK de transmisión de IVS | Transmisión en tiempo real

El SDK de transmisión para transmisión en tiempo real de Amazon Interactive Video Service (IVS) está pensado para desarrolladores que crean aplicaciones con Amazon IVS. Este SDK está diseñado a fin de aprovechar la arquitectura de Amazon IVS y verá mejoras continuas y nuevas características, junto con Amazon IVS. Como SDK de transmisión nativo, está diseñado para minimizar el impacto en el rendimiento de la aplicación y en los dispositivos con los que los usuarios acceden a la aplicación.

Tenga en cuenta que el SDK de transmisión se usa tanto para enviar como para recibir videos; es decir, utiliza el mismo SDK para los hosts y para los espectadores. No se necesita el SDK de un reproductor independiente.

Su aplicación puede aprovechar las características clave del SDK de transmisión de Amazon IVS:

- **Transmisión de alta calidad:** el SDK de transmisión admite la transmisión de alta calidad. Reciba video de su cámara y codifíquelo a una velocidad de hasta 720p.
- **Ajustes automáticos de la velocidad de bits:** los usuarios de smartphones son móviles, por lo que sus condiciones de red pueden cambiar a lo largo de una transmisión. El SDK de transmisión de Amazon IVS ajusta automáticamente la velocidad de bits de video para adaptarse a las condiciones cambiantes de la red.
- **Soporte vertical y horizontal:** independientemente del modo en que los usuarios mantengan sus dispositivos, la imagen aparece con el lado correcto hacia arriba y se escala según corresponda. El SDK de transmisión admite el tamaño de formato vertical y horizontal. Administra automáticamente la relación de aspecto cuando los usuarios rotan su dispositivo hacia una orientación distinta de la configurada.
- **Transmisión segura:** las transmisiones de su usuario se cifran mediante TLS, por lo que pueden mantener sus transmisiones seguras.
- **Dispositivos de audio externos:** el SDK de transmisión de Amazon IVS admite micrófonos externos SCO de audio, USB y Bluetooth.

Requisitos de la plataforma

Plataformas nativas

Plataforma	Versiones compatibles
Android	9.0 y versiones posteriores: tenga en cuenta que los clientes pueden crear con la versión 5.0, pero no podrán utilizar la funcionalidad de transmisión en tiempo real.
iOS	14 y versiones posteriores

IVS admite un mínimo de 4 versiones principales de iOS y 6 versiones principales de Android. El soporte de nuestra versión actual se puede extender más allá de estos mínimos. Los clientes recibirán una notificación mediante las notas de la versión del SDK con al menos 3 meses de antelación cuando una versión principal deje de ser compatible.

Navegadores de escritorio

Navegador	Plataformas admitidas	Versiones compatibles
Chrome	Windows, macOS	Dos versiones principales (la versión actual y la anterior más reciente)
Firefox	Windows, macOS	Dos versiones principales (la versión actual y la anterior más reciente)
Ubicaciones	Windows 8.1 y posteriores	Dos versiones principales (la versión actual y la anterior más reciente) No incluye Edge Legacy
Safari	macOS	Dos versiones principales (la versión actual y la anterior más reciente)

Navegadores móviles (iOS y Android)

Navegador	Plataformas admitidas	Versiones compatibles
Chrome	iOS, Android	Dos versiones principales (la versión actual y la anterior más reciente)
Firefox	Android	Dos versiones principales (la versión actual y la anterior más reciente)
Safari	iOS	Dos versiones principales (la versión actual y la anterior más reciente)

Limitaciones conocidas

- En todos los dispositivos móviles, no recomendamos publicar o suscribirse con cuatro o más participantes al mismo tiempo, debido a problemas con los artefactos de vídeo y las pantallas negras. Si necesita más participantes, configure la opción de [publicar y suscribirse solo en audio](#).
- No recomendamos componer un escenario y retransmitirlo a un canal de Android Mobile Web por motivos de rendimiento y posibles bloqueos. Si se requiere la funcionalidad de transmisión, integre el [SDK de transmisión para transmisión en tiempo real de IVS para Android](#).

Vistas web

El SDK de transmisión web no admite vistas web ni entornos similares a los de la web (televisores, consolas, etc.). Para implementaciones móviles, consulte la Guía del SDK de transmisión de transmisión en tiempo para [Android](#) y para [iOS](#).

Se requiere acceso a los dispositivos

El SDK de difusión requiere acceso a las cámaras y micrófonos del dispositivo, tanto los integrados en el dispositivo como los conectados a través de Bluetooth, USB o conector de audio.

Soporte

El SDK de transmisión se mejora de forma continua. Consulte [Notas de la versión de Amazon IVS](#) para ver las versiones disponibles y los problemas solucionados. Si procede, antes de contactar con el soporte técnico, actualice su versión del SDK de transmisión y compruebe si se resuelve el problema.

Control de versiones

Los SDK de transmisión de Amazon IVS utilizan el [control de versiones semántico](#).

Para este análisis, suponga:

- La última versión es la 4.1.3.
- La última versión de la versión principal anterior es la 3.2.4.
- La última versión de la versión 1.x es la 1.5.6.

Las características nuevas compatibles con versiones anteriores se agregan como versiones secundarias de la última versión. En este caso, el siguiente conjunto de características nuevas se agregará como la versión 4.2.0.

Se agregan correcciones de errores menores compatibles con versiones anteriores como parches de la última versión. Aquí, el siguiente conjunto de correcciones de errores menores se agregará como la versión 4.1.4.

Las correcciones de errores principales compatibles con versiones anteriores se manejan de manera diferente; estas se agregan a varias versiones:

- Versión del parche de la última versión. Aquí, esta es la versión 4.1.4.
- Versión del parche de la versión secundaria anterior. Aquí, esta es la versión 3.2.5.
- Versión del parche de la última versión 1.x. Aquí, esta es la versión 1.5.7.

El equipo de productos de Amazon IVS define las principales correcciones de errores. Las actualizaciones de seguridad críticas y otras correcciones seleccionadas necesarias para los clientes son ejemplos típicos.

Nota: En los ejemplos anteriores, las versiones publicadas aumentan sin omitir ningún número (por ejemplo, de 4.1.3 a 4.1.4). En realidad, uno o más números de parche pueden permanecer internos y no ser lanzados, por lo que la versión publicada podría aumentar de 4.1.3 a 4.1.6.

SDK de transmisión de IVS: guía para web | Transmisión en tiempo real

El SDK de transmisión web para transmisión en tiempo real de IVS ofrece a los desarrolladores las herramientas para crear experiencias interactivas y en tiempo real en la web. El SDK está pensado para los desarrolladores que crean aplicaciones web con Amazon IVS.

El SDK de transmisión web permite a los participantes enviar y recibir videos. El SDK admite las siguientes operaciones:

- Incorporación a un escenario
- Publicación de contenido multimedia para otros participantes del escenario
- Suscripción al contenido multimedia de otros participantes del escenario
- Administración y monitoreo del video y audio publicados en el escenario
- Obtención de estadísticas de WebRTC de cada conexión de pares
- Todas las operaciones del SDK de transmisión web de transmisión de baja latencia de IVS

Versión más reciente del SDK de transmisión para web: 1.17.0 ([notas de la versión](#))

Documentación de referencia: a fin de obtener información sobre los métodos más importantes disponibles en el SDK de transmisión web de Amazon IVS, consulte <https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference>. Asegúrese de seleccionar la versión más reciente del SDK.

Código de ejemplo: los siguientes ejemplos son un buen punto para empezar rápidamente a usar el SDK:

- [HTML y JavaScript](#)
- [React](#)

Requisitos de la plataforma: consulte [SDK de transmisión de Amazon IVS](#) para obtener un listado de las plataformas compatibles.

Introducción al SDK de transmisión para web de IVS | Transmisión en tiempo real

Este documento explica los pasos necesarios para comenzar a utilizar el SDK de transmisión para web de la transmisión en tiempo real de IVS.

Importaciones

Los componentes básicos de la transmisión en tiempo real se encuentran en un espacio de nombres diferente al de los módulos de transmisión raíz.

Uso de una etiqueta de script

Con las mismas importaciones de scripts, las clases y enumeraciones definidas en los ejemplos siguientes se pueden encontrar en el objeto global `IVSBroadcastClient`:

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

Con npm

Las clases, enumeraciones y tipos también se pueden importar desde el módulo del paquete:

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

Compatibilidad con la representación del servidor

La biblioteca de fases del SDK de transmisión para web no se puede cargar en un contexto del servidor, ya que hace referencia a los elementos básicos del navegador necesarios para el funcionamiento de la biblioteca cuando se carga. Para solucionar este problema, cargue la biblioteca de forma dinámica, como se muestra en la [demostración de transmisión web con Next y React](#).

Solicitar permisos

La aplicación debe solicitar permiso para acceder a la cámara y al micrófono del usuario, y deberán ser ofrecidos mediante HTTPS. (Esto no es específico de Amazon IVS; es necesario para cualquier sitio web que necesite acceso a cámaras y micrófonos).

A continuación, le presentamos una función a modo de ejemplo que muestra cómo solicitar y capturar permisos para los dispositivos de audio y 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.');
```

Para obtener más información, consulte la [API de permisos](#) y [MediaDevices.getUserMedia\(\)](#).

Enlistar los dispositivos disponibles

Para conocer los dispositivos que puede capturar, consulte el método [MediaDevices.enumerateDevices\(\)](#) del navegador:

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

Recuperar MediaStream de un dispositivo

Después de obtener la lista de dispositivos disponibles, puede recuperar una transmisión de diversos dispositivos. Por ejemplo, puede utilizar el método `getUserMedia()` para recuperar la transmisión de una cámara.

Si quisiera indicar desde cuál dispositivo desea capturar la transmisión, puede establecer de forma expresa `deviceId` en las secciones `audio` o `video` de las restricciones de multimedia. De forma alternativa, puede omitir `deviceId` y que los usuarios seleccionen los dispositivos desde el símbolo del navegador.

También puede especificar la resolución de cámara ideal mediante las restricciones `width` y `height`. (Obtenga más información sobre estas restricciones [aquí](#)). El SDK aplica de forma automática las restricciones máximas de ancho y alto que corresponden a la resolución máxima de transmisión; sin embargo, es aconsejable que las aplique usted mismo para garantizar que la relación de aspecto de la fuente no cambie después de que la agregue al SDK.

Para la transmisión en tiempo real, asegúrese de que el contenido multimedia esté limitado a una resolución de 720p. En concreto, sus valores de ancho y alto de la restricción de `getUserMedia` y `getDisplayMedia` no deben ser superiores a 921 600 (1280*720) cuando se multiplican.

```
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 },
});
```

Publicación y suscripción con el SDK de transmisión para web de IVS | Transmisión en tiempo real

Este documento explica los pasos necesarios para publicar y suscribirse a una fase mediante el SDK de transmisión para web de la transmisión en tiempo real de IVS.

Conceptos

Los siguientes tres conceptos básicos subyacen a la funcionalidad de transmisión en tiempo real: [escenario](#), [estrategia](#) y [eventos](#). El objetivo del diseño es minimizar la cantidad de lógica necesaria por parte del cliente para crear un producto que funcione.

Escenario

La clase Stage es el principal punto de interacción entre la aplicación host y el SDK. Representa el escenario como tal y se usa para entrar y salir de él. Para crear un escenario e incorporarse a él, es necesaria una cadena de símbolos válida y que no haya vencido del plano de control (representada como token). Entrar y salir de un escenario es sencillo:

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

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

stage.leave();
```

Strategy (Estrategia)

La interfaz StageStrategy proporciona una forma para que la aplicación host comunique el estado deseado del escenario al SDK. Es necesario implementar las siguientes tres funciones: `shouldSubscribeToParticipant`, `shouldPublishParticipant` y `stageStreamsToPublish`. Todas se analizan a continuación.

Para usar una estrategia definida, pásela al constructor de Stage. El siguiente es un ejemplo completo de una aplicación que utiliza una estrategia para publicar la cámara web de un participante en el escenario y suscribirse a todos los participantes. El propósito de cada función de estrategia necesaria se explica detalladamente en las siguientes secciones.

```
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();
```

Suscripción a participantes

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

Cuando un participante remoto se incorpora al escenario, el SDK consulta la aplicación host sobre el estado de suscripción deseado de ese participante. Las opciones son NONE, AUDIO_ONLY y AUDIO_VIDEO. Al devolver un valor para esta función, la aplicación host no tiene que preocuparse por el estado de la publicación, el estado actual de la suscripción ni el estado de la conexión del escenario. Si se devuelve AUDIO_VIDEO, el SDK espera a que el participante remoto publique antes de suscribirse y actualiza la aplicación host al emitir eventos durante todo el proceso.

Este es un ejemplo de implementación:

```
const strategy = {  
  
  shouldSubscribeToParticipant: (participant) => {  
    return SubscribeType.AUDIO_VIDEO;  
  }  
  
  // ... other strategy functions  
}
```

Esta es la implementación completa de esta función para una aplicación host que siempre quiere que todos los participantes se vean entre sí; por ejemplo, una aplicación de videochat.

También son posibles implementaciones más avanzadas. Por ejemplo, supongamos que la aplicación proporciona un atributo `role` al crear el token con `CreateParticipantToken`. La aplicación podría utilizar la propiedad `attributes` en `StageParticipantInfo` para suscribirse de forma selectiva a los participantes en función de los atributos proporcionados por el servidor:

```
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  
}
```

Esto se puede utilizar para crear un escenario en el que los moderadores puedan monitorear a todos los invitados sin que ellos mismos los vean ni los escuchen. La aplicación host podría utilizar una lógica empresarial adicional para permitir que los moderadores se vean entre sí, pero permanezcan invisibles para los invitados.

Configuración de la suscripción a participantes

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

Si se está suscribiendo a un participante remoto (consulte [Suscripción a participantes](#)), el SDK consulta a la aplicación host sobre una configuración de la suscripción personalizada para ese participante. Esta configuración es opcional y permite a la aplicación host controlar determinados aspectos del comportamiento de los suscriptores. Para obtener información sobre lo que se puede configurar, consulte [SubscribeConfiguration](#) en la documentación de referencia del SDK.

Este es un ejemplo de implementación:

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

Esta implementación actualiza el retraso mínimo del búfer de fluctuación para todos los participantes suscritos al valor preestablecido MEDIUM.

Como en el caso de `shouldSubscribeToParticipant`, también son posibles implementaciones más avanzadas. El valor de `ParticipantInfo` proporcionado se puede utilizar para actualizar selectivamente la configuración de suscripción para participantes específicos.

Se recomienda utilizar los comportamientos predeterminados. Especifique la configuración personalizada solo si hay un comportamiento en particular que desee cambiar.

Publicación

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

Una vez realizada la conexión al escenario, el SDK consulta la aplicación host para ver si un participante en particular tiene que publicar. Esto solo se invoca en los participantes locales que tienen permiso para publicar en función del token proporcionado.

Este es un ejemplo de implementación:

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

Esto es para una aplicación de videochat estándar en la que los usuarios siempre quieren publicar. Pueden activar y desactivar el sonido y el video para ocultarse o verse y escucharse al instante. (También pueden usar la opción de publicar o anular la publicación, pero esto es mucho más lento. Es preferible silenciar o activar el sonido en casos de uso en los que se quiera cambiar la visibilidad de manera frecuente).

Elección de las transmisiones que publicar

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

Al publicar, esto se utiliza para determinar qué transmisiones de audio y video se tienen que publicar. Esto se explica en mayor detalle más adelante en [Publicación de una transmisión multimedia](#).

Actualización de la estrategia

La estrategia pretende ser dinámica: los valores devueltos por cualquiera de las funciones anteriores se pueden cambiar en cualquier momento. Por ejemplo, si la aplicación host no quiere publicar hasta que el usuario final presione un botón, puede devolver una variable de `shouldPublishParticipant` (como `hasUserTappedPublishButton`). Cuando esa variable cambie en función de una interacción del usuario final, llame a `stage.refreshStrategy()` para indicar al SDK que debe consultar la estrategia a fin de obtener los valores más recientes y aplicar

solo los cambios. Si el SDK observa que el valor `shouldPublishParticipant` cambió, se iniciará el proceso de publicación. Si las consultas del SDK y todas las funciones devuelven el mismo valor que antes, la llamada a `refreshStrategy` no hará cambios en el escenario.

Si el valor devuelto de `shouldSubscribeToParticipant` cambia de `AUDIO_VIDEO` a `AUDIO_ONLY`, la transmisión de video se elimina para todos los participantes con valores devueltos modificados, si ya existía con anterioridad una transmisión de video.

Por lo general, el escenario utiliza la estrategia para aplicar de la manera más eficiente la diferencia entre las estrategias anteriores y actuales, sin que la aplicación host tenga que preocuparse por todo el estado necesario para administrarla correctamente. Por eso, piense en la llamada a `stage.refreshStrategy()` como una operación barata, porque no hace nada a no ser que cambie la estrategia.

Eventos

Una instancia `Stage` es un emisor de eventos. Con `stage.on()`, el estado del escenario se comunica a la aplicación host. Por lo general, los eventos permiten actualizar por completo la interfaz de usuario de la aplicación host. Los eventos son los siguientes:

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

Para la mayoría de estos métodos, se proporciona la `ParticipantInfo` correspondiente.

No se espera que la información que proporcionan los eventos afecte a los valores de retorno de la estrategia. Por ejemplo, no se espera que el valor devuelto de `shouldSubscribeToParticipant` cambie cuando se llama a `STAGE_PARTICIPANT_PUBLISH_STATE_CHANGED`. Si la aplicación host quiere suscribirse a un participante en particular, debe devolver el tipo de suscripción deseado, independientemente del estado de publicación de ese participante. El SDK es responsable de garantizar que se aplique el estado deseado de la estrategia en el momento correcto según el estado del escenario.

Publicación de una transmisión multimedia

Los dispositivos locales, como micrófonos y cámaras, se recuperan siguiendo los mismos pasos descritos anteriormente en [Recuperación de MediaStream de un dispositivo](#). En el ejemplo, utilizamos `MediaStream` para crear una lista de objetos `LocalStageStream` que el SDK utiliza para publicar:

```
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
  }
}
```

Publicación de una pantalla compartida

Las aplicaciones suelen necesitar publicar una pantalla compartida además de la cámara web del usuario. Para publicar una pantalla compartida, es necesario crear un token adicional para la fase, específicamente para publicar el contenido multimedia de la pantalla compartida. Use `getDisplayMedia` y restrinja la resolución a un máximo de 720p. Después de eso, los pasos son similares a publicar una cámara en la fase.

```
// 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();
```

Visualización y eliminación de participantes

Cuando se complete la suscripción, recibirá una matriz de objetos `StageStream` a través del evento `STAGE_PARTICIPANT_STREAMS_ADDED`. El evento también le brinda información sobre los participantes que le será de ayuda al visualizar las transmisiones multimedia:

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

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

  // Create or find video element already available in your application
  const videoEl = getParticipantVideoElement(participant.id);

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

Cuando un participante deja de publicar o anula la suscripción de una transmisión, se invoca la función `STAGE_PARTICIPANT_STREAMS_REMOVED` con las transmisiones que se eliminaron. Las aplicaciones host tienen que usar esto como una señal para eliminar la transmisión de video del participante de DOM.

STAGE_PARTICIPANT_STREAMS_REMOVED se invoca para todas las situaciones en las que se puede eliminar una transmisión, como las siguientes:

- El participante remoto deja de publicar.
- Un dispositivo local cancela la suscripción o cambia la suscripción de AUDIO_VIDEO a AUDIO_ONLY.
- El participante remoto abandona el escenario.
- El participante local abandona el escenario.

Ya que STAGE_PARTICIPANT_STREAMS_REMOVED se invoca en todas las situaciones, no es necesaria una lógica empresarial personalizada para eliminar a los participantes de la interfaz de usuario durante las operaciones de licencia remota o local.

Activación y desactivación del sonido de las transmisiones multimedia

Los objetos de `LocalStageStream` tienen una función `setMuted` que controla si la transmisión está silenciada. Esta función se puede invocar en la transmisión antes o después de que la devuelva la función de estrategia `stageStreamsToPublish`.

Importante: Si `stageStreamsToPublish` devuelve una nueva instancia de objeto de `LocalStageStream` después de una llamada a `refreshStrategy`, el estado de sonido desactivado del nuevo objeto de transmisión se aplicará al escenario. Tenga cuidado al crear nuevas instancias `LocalStageStream` para asegurarse de que se mantenga el estado de sonido desactivado que se espera.

Monitoreo del estado de sonido desactivado en los contenidos multimedia del participante remoto

Cuando los participantes cambian el estado de sonido desactivado de su video o audio, el evento `STAGE_STREAM_MUTE_CHANGED` se activa con una lista de las transmisiones que cambiaron. Use la propiedad `isMuted` en `StageStream` para actualizar su interfaz de usuario según corresponda:

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

Además, puede consultar [StageParticipantInfo](#) para obtener información del estado sobre si el audio o el video están silenciados:

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

Obtención de estadísticas de WebRTC

Para obtener las estadísticas más recientes de WebRTC de una transmisión de publicación o de suscripción, utilice `getStats` en `StageStream`. Este es un método asíncrono con el que puede recuperar estadísticas mediante `await` o encadenando una promesa. El resultado es un `RTCStatsReport`, un diccionario que contiene todas las estadísticas estándar.

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

Optimización del contenido multimedia

Se recomienda limitar las llamadas a `getUserMedia` y a `getDisplayMedia` con las siguientes restricciones para obtener el mejor rendimiento:

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

Puede restringir aún más el contenido multimedia mediante opciones adicionales que se pasan al constructor `LocalStageStream`:

```
const localStreamOptions = {
```

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

En el código anterior:

- `minBitrate` establece la velocidad de bits mínima que se espera que utilice el navegador. Sin embargo, una transmisión de video de poca complejidad puede hacer que el codificador establezca una velocidad inferior a esta velocidad de bits.
- `maxBitrate` establece una velocidad de bits máxima que se espera que el navegador no supere para esta transmisión.
- `maxFramerate` establece una velocidad de fotogramas máxima que se espera que el navegador no supere para esta transmisión.
- La opción `simulcast` solo se puede utilizar en navegadores basados en Chromium. Permite enviar tres capas de representación de la transmisión.
 - Esto permite al servidor elegir qué representación enviar a otros participantes, en función de sus limitaciones de red.
 - Cuando `simulcast` se especifica junto con un valor de `maxBitrate` o `maxFramerate`, se espera que la capa de representación más alta se configure teniendo en cuenta estos valores, siempre que `maxBitrate` no sea inferior al valor predeterminado de `maxBitrate` (900 kbps) de la segunda capa más alta del SDK interno.
 - Si `maxBitrate` se especifica como demasiado baja en comparación con el valor predeterminado de la segunda capa más alta, `simulcast` se desactivará.
 - `simulcast` no se puede activar y desactivar sin volver a publicar el contenido multimedia. Para ello, `shouldPublishParticipant` tiene que devolver `false` y llamar a `refreshStrategy`; `shouldPublishParticipant` tiene que devolver `true` y llamar a `refreshStrategy` otra vez.

Obtención de los atributos de los participantes

Si especifica atributos en la solicitud del punto de conexión de `CreateParticipantToken`, puede ver los atributos en las propiedades de `StageParticipantInfo`:

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

Gestión de los problemas de red

Cuando se pierde la conexión de red del dispositivo local, el SDK se intenta volver a conectar internamente sin que el usuario lleve a cabo ninguna acción. En algunos casos, el SDK no funciona de manera correcta y es necesario que el usuario actúe.

En términos generales, el estado del escenario se puede gestionar mediante el evento `STAGE_CONNECTION_STATE_CHANGED`:

```
stage.on(StageEvents.STAGE_CONNECTION_STATE_CHANGED, (state) => {
  switch (state) {
    case StageConnectionState.DISCONNECTED:
      // handle disconnected UI
      break;
    case StageConnectionState.CONNECTING:
      // handle establishing connection UI
      break;
    case StageConnectionState.CONNECTED:
      // SDK is connected to the Stage
      break;
    case StageConnectionState.ERRORRED:
      // SDK encountered an error and lost its connection to the stage. Wait for
      CONNECTED.
      break;
  }
})
```

En general, puede ignorar un estado de error que se produzca tras unirse correctamente a una fase, ya que el SDK intentará recuperarse internamente. Si el SDK informa de un estado `ERRORRED` y la fase permanece en el estado `CONNECTING` durante un periodo prolongado (por ejemplo, 30 segundos o más), es probable que se haya desconectado de la red.

Transmisión del escenario a un canal de IVS

Para transmitir un escenario, cree una sesión de `IVSBroadcastClient` independiente y, a continuación, siga las instrucciones habituales para la transmisión con el SDK, descritas con anterioridad. La lista de `StageStream` expuestas mediante

STAGE_PARTICIPANT_STREAMS_ADDED se puede utilizar para recuperar las transmisiones multimedia participantes aplicables a la composición del flujo de transmisión, de la siguiente manera:

```
// 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;
    }
  })
})
```

Si lo desea, puede componer un escenario y transmitirlo a un canal de IVS de baja latencia para llegar a un público más amplio. Consulte [Habilitación de varios hosts en una transmisión de Amazon IVS](#) en la Guía del usuario de transmisión de baja latencia.

Problemas conocidos y soluciones alternativas del SDK de transmisión para web de IVS | Transmisión en tiempo real

Este documento enumera problemas conocidos que puede experimentar al utilizar el SDK de transmisión para web de la transmisión en tiempo real de Amazon IVS y sugiere posibles soluciones alternativas.

- Al cerrar las pestañas del navegador o salir de los navegadores sin llamar a `stage.leave()`, los usuarios pueden seguir apareciendo en la sesión con una pantalla congelada o negra durante un máximo de 10 segundos.

Solución alternativa: ninguna.

- Las sesiones de Safari aparecen de forma intermitente con una pantalla negra para que los usuarios se unan una vez iniciada la sesión.

Solución alternativa: actualice el navegador y vuelva a conectar la sesión.

- Safari no se recupera correctamente al cambiar de red.

Solución alternativa: actualice el navegador y vuelva a conectar la sesión.

- La consola para desarrolladores repite un error `Error: UnintentionalError at StageSocket.onClose`.

Solución alternativa: solo se puede crear un escenario por token de participante. Este error se produce cuando se crea más de una instancia Stage con el mismo token de participante, independientemente de si la instancia está en un dispositivo o en varios.

- Es posible que tenga problemas para mantener el estado `StageParticipantPublishState.PUBLISHED` y que reciba estados `StageParticipantPublishState.ATTEMPTING_PUBLISH` repetidos al escuchar el evento `StageEvents.STAGE_PARTICIPANT_PUBLISH_STATE_CHANGED`.

Solución alternativa: limite la resolución de video a 720p al invocar a `getUserMedia` o `getDisplayMedia`. En concreto, sus valores de ancho y alto de la restricción de `getUserMedia` y `getDisplayMedia` no deben ser superiores a 921 600 (1280*720) cuando se multiplican.

Limitaciones de Safari

- Para denegar un mensaje de permisos, debe restablecer el permiso en la configuración del sitio web de Safari en el sistema operativo.
- Safari no detecta todos los dispositivos de forma directa con la misma eficacia que Firefox o Chrome. Por ejemplo, no identifica la cámara virtual OBS.

Limitaciones de Firefox

- Los permisos del sistema deben estar habilitados para que Firefox pueda compartir la pantalla. Luego de activarlos, el usuario debe reiniciar Firefox para que funcione sin inconvenientes; de lo contrario, si considera que los permisos están bloqueados, el navegador generará una excepción [NotFoundError](#).

- Falta el método `getCapabilities`. Esto significa que los usuarios no pueden obtener la resolución o la relación de aspecto de la pista multimedia. Consulte este [hilo de Bugzilla](#).
- Faltan varias propiedades `AudioContext`, por ejemplo, la latencia y el recuento de canales. Esto podría suponer un problema para los usuarios avanzados que desean manejar las pistas de audio.
- Las imágenes de la cámara de `getUserMedia` están limitadas a una relación de aspecto 4:3 en MacOS. Consulte el [hilo 1 de Bugzilla](#) y el [hilo 2 de Bugzilla](#).
- Con `getDisplayMedia`, la captura de audio no es compatible. Consulte este [hilo de Bugzilla](#).
- La velocidad de fotogramas en la captura de pantalla es poco óptima (¿aproximadamente 15 fps?). Consulte este [hilo de Bugzilla](#).

Limitaciones de la web móvil

- Los dispositivos móviles no admiten el uso compartido de pantalla con [getDisplayMedia](#).

Solución alternativa: ninguna.

- El participante tarda entre 15 y 30 segundos en salir cuando cierra un navegador sin llamar a `leave()`.

Solución alternativa: añada una interfaz de usuario que anime a los usuarios a desconectarse correctamente.

- Poner en segundo plano la aplicación hace que se detenga la publicación del vídeo.

Solución alternativa: muestre una lista de interfaz de usuario cuando el publicador esté en pausa.

- La velocidad de fotogramas del vídeo se reduce durante aproximadamente 5 segundos después de desactivar el silenciamiento de una cámara en los dispositivos Android.

Solución alternativa: ninguna.

- La transmisión de vídeo se estira al girar para iOS 16.0.

Solución alternativa: muestre una interfaz de usuario en la que se describa este problema conocido del sistema operativo.

- Al cambiar el dispositivo de entrada de audio, se cambia automáticamente el dispositivo de salida de audio.

Solución alternativa: ninguna.

- Al poner en segundo plano el navegador, el flujo de publicación se pone en negro y solo produce audio.

Solución alternativa: ninguna. Lo hacemos por motivos de seguridad.

Control de errores en el SDK de transmisión para web de IVS | Transmisión en tiempo real

En esta sección se ofrece información general sobre las condiciones de error, cómo el SDK de transmisión para web las notifica a la aplicación y qué debe hacer una aplicación cuando se producen esos errores. El SDK informa de los errores a los oyentes del evento `StageEvents.ERROR`:

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

Errores de fases

Se informa de un `StageError` cuando el SDK encuentra un problema del que no puede recuperarse y, por lo general, requiere la intervención de la aplicación o la reconexión de la red para recuperarse.

Cada `StageError` notificado tiene un código (o `StageErrorCode`), un mensaje (cadena) y una categoría (`StageErrorCategory`). Cada uno está relacionado con una categoría de operación subyacente.

La categoría de operación del error se determina en función de si está relacionado con la conexión a la fase (`JOIN_ERROR`), el envío de contenido multimedia a la fase (`PUBLISH_ERROR`) o la recepción de una transmisión de contenido multimedia entrante desde la fase (`SUBSCRIBE_ERROR`).

La propiedad de código de un `StageError` notifica el problema específico:

Nombre	Código	Acción recomendada
TOKEN_MALFORMED	1	Cree un token válido y vuelva a intentar crear instancias para la fase.

Nombre	Código	Acción recomendada
TOKEN_EXPIRED	2	Cree un token no vencido y vuelva a intentar crear instancias para la fase.
TIMEOUT	3	La operación ha agotado el tiempo de espera. Si la fase existe y el token es válido, es probable que este error se deba a un problema de red. En ese caso, espere a que se recupere la conectividad del dispositivo.
ERROR	4	<p>Se produjo una condición grave al intentar completar una operación. Compruebe los detalles del error.</p> <p>Si la fase existe y el token es válido, es probable que este error se deba a un problema de red. En ese caso, espere a que se recupere la conectividad del dispositivo.</p>
CANCELADO	5	Compruebe el código de la aplicación y asegúrese de que no haya invocaciones de <code>join</code> , <code>refreshStrategy</code> o <code>replaceStrategy</code> repetidas, ya que puede provocar que las operaciones repetidas se inicien y cancelen antes de completarse.
STAGE_AT_CAPACITY	6	Vuelva a intentar la operación cuando la fase ya no esté llena mediante la actualización de la estrategia.
CODEC_MISMATCH	7	La fase no admite el códec. Compruebe si el navegador y la plataforma son compatibles con el códec. Para la transmisión en tiempo real de IVS, los navegadores deben admitir el códec H.264 para video y el códec Opus para audio.
TOKEN_NOT_ALLOWED	8	El token no tiene permiso para la operación. Vuelva a crear el token con los permisos correctos e inténtelo de nuevo.

Ejemplo de gestión de StageError

Use el código StageError para determinar si el error se debe a un token vencido:

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

Errores de red cuando ya está conectado

Si se interrumpe la conexión de red del dispositivo, es posible que el SDK pierda la conexión con los servidores de fases. Es posible que observe errores en la consola porque el SDK ya no puede acceder a los servicios de backend. Las publicaciones en <https://broadcast.stats.live-video.net> fallarán.

Si está publicando o suscribiéndose, observará errores en la consola relacionados con los intentos de publicar/suscribirte.

Internamente, el SDK intentará reconectarse con una estrategia de retroceso exponencial.

Acción: espere a que se recupere la conectividad del dispositivo.

Estados de error

Le recomendamos que utilice estos estados para el registro de las aplicaciones y para mostrar mensajes a los usuarios que les avisen de problemas de conectividad con la fase para un participante concreto.

Publicación

El SDK informa ERRORRED cuando se produce un error en una publicación.

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

Suscribirse

El SDK informa `ERRORRED` cuando se produce un error en la suscripción. Esto puede ocurrir debido a las condiciones de la red o si una etapa está llena para los suscriptores.

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

SDK de transmisión de IVS: guía para Android | Transmisión en tiempo real

El SDK de transmisión para transmisión en tiempo real de IVS para Android permite a los participantes enviar y recibir videos en Android.

El paquete de `com.amazonaws.ivs.broadcast` implementa la interfaz descrita en este documento. El SDK admite las siguientes operaciones:

- Incorporación a un escenario
- Publicación de contenido multimedia para otros participantes del escenario
- Suscripción al contenido multimedia de otros participantes del escenario
- Administración y monitoreo del video y audio publicados en el escenario
- Obtención de estadísticas de WebRTC de cada conexión de pares
- Todas las operaciones del SDK de transmisión para Android de transmisión de baja latencia de IVS

Versión más reciente del SDK de transmisión para Android: 1.23.0 ([notas de la versión](#))

Documentación de referencia: a fin de obtener información sobre los métodos más importantes disponibles en el SDK de transmisión de Android de Amazon IVS, consulte la documentación de referencia en <https://aws.github.io/amazon-ivs-broadcast-docs/1.23.0/android/>.

Código de muestra: consulte el repositorio de muestra de Android en GitHub: <https://github.com/aws-samples/amazon-ivs-broadcast-android-sample>.

Requisitos de la plataforma: Android 9.0 y versiones posteriores.

Introducción al SDK de transmisión para Android de IVS | Transmisión en tiempo real

Este documento explica los pasos necesarios para comenzar a utilizar el SDK de transmisión para Android de la transmisión en tiempo real de IVS.

Instalación de la biblioteca

Hay varias formas de agregar la biblioteca de transmisión para Android de Amazon IVS a su entorno de desarrollo de Android: uso directo de Gradle, uso de los catálogos de versiones de Gradle o instalación manual del SDK.

Uso directo de Gradle: agregue la biblioteca al archivo `build.gradle` del módulo, como se muestra a continuación (para la versión más reciente del SDK de transmisión de IVS):

```
repositories {
    mavenCentral()
}

dependencies {
    implementation 'com.amazonaws:ivs-broadcast:1.23.0:stages@aar'
}
```

Uso de los catálogos de versiones de Gradle: primero incluya esto en el archivo `build.gradle` del módulo:

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

A continuación, incluya lo siguiente en el archivo `libs.version.toml` (para obtener la versión más reciente del SDK de transmisión de IVS):

```
[versions]
ivs="1.23.0"
```

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

Instalación manual del SDK: descargue la versión más reciente desde esta ubicación:

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

Asegúrese de descargar el archivo aar con `-stages` adjunto.

Permita también el control de SDK a través del altavoz: independientemente del método de instalación que elija, agregue también el siguiente permiso al manifiesto para permitir que el SDK habilite o deshabilite el altavoz:

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

Uso del SDK con símbolos de depuración

También publicamos una versión del SDK de transmisión para Android que incluye símbolos de depuración. Puede usar esta versión para mejorar la calidad de los informes de depuración (seguimientos de pila) en Firebase Crashlytics si se produce algún fallo en el SDK de transmisión de IVS, es decir, `libbroadcastcore.so`. Cuando notifica estos bloqueos al equipo del SDK de IVS, los rastreos de pila de mayor calidad facilitan la solución de los problemas.

Para usar esta versión del SDK, coloque lo siguiente en los archivos de compilación de Gradle:

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

Use la línea anterior en lugar de esta:

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

Carga de símbolos en Firebase Crashlytics

Asegúrese de que los archivos de compilación de Gradle estén configurados para Firebase Crashlytics. Siga las instrucciones de Google aquí:

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

Asegúrese de incluir `com.google.firebase:firebase-crashlytics-ndk` como dependencia.

Al crear la aplicación para su lanzamiento, el complemento de Firebase Crashlytics debería cargar los símbolos automáticamente. Para cargar los símbolos manualmente, ejecute cualquiera de los comandos siguientes:

```
gradle uploadCrashlyticsSymbolFileRelease
```

```
./gradlew uploadCrashlyticsSymbolFileRelease
```

(No pasará nada si los símbolos se cargan dos veces, automática y manualmente).

Cómo evitar que el archivo .apk de la versión aumente de tamaño

Antes de empaquetar el archivo .apk de la versión, el complemento de Gradle para Android intenta eliminar automáticamente la información de depuración de las bibliotecas compartidas (incluida la biblioteca `libbroadcastcore.so` del SDK de transmisión de IVS). Sin embargo, a veces esto no sucede. Como resultado, el archivo .apk podría aumentar de tamaño y podría recibir un mensaje de advertencia del complemento de Gradle para Android indicándole que no puede eliminar los símbolos de depuración y que está empaquetando los archivos .so tal como están. Si esto sucede, haga lo siguiente:

- Instale un NDK de Android. Cualquier versión reciente funcionará.
- Agregue `ndkVersion <your_installed_ndk_version_number>` al archivo `build.gradle` de la aplicación. Haga esto incluso si la propia aplicación no contiene código nativo.

Para obtener más información, consulte este [informe de problemas](#).

Solicitar permisos

La aplicación debe solicitar permiso para acceder a la cámara y al micrófono del usuario. (Esto no es específico de Amazon IVS; es necesario para cualquier aplicación que necesite acceso a cámaras y micrófonos).

Aquí, verificamos si el usuario ya ha concedido permisos y, de no ser así, preguntamos por ellos:

```
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;
}
}
```

Aquí, obtenemos la respuesta del usuario:

```
@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();
    }
}
```

Publicación y suscripción con el SDK de transmisión para Android de IVS | Transmisión en tiempo real

Este documento explica los pasos necesarios para publicar y suscribirse a una fase mediante el SDK de transmisión para Android de la transmisión en tiempo real de IVS.

Conceptos

Los siguientes tres conceptos básicos subyacen a la funcionalidad de transmisión en tiempo real: [escenario](#), [estrategia](#) y [renderizador](#). El objetivo del diseño es minimizar la cantidad de lógica necesaria por parte del cliente para crear un producto que funcione.

Escenario

La clase Stage es el principal punto de interacción entre la aplicación host y el SDK. Representa el escenario como tal y se usa para entrar y salir de él. Para crear un escenario e incorporarse a él, es

necesaria una cadena de símbolos válida y que no haya vencido del plano de control (representada como token). Entrar a un escenario y salir de él es sencillo.

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

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

stage.leave();
```

También se puede adjuntar `StageRenderer` en la clase `Stage`:

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

Strategy (Estrategia)

La interfaz `Stage.Strategy` proporciona una forma para que la aplicación host comunique el estado deseado del escenario al SDK. Es necesario implementar las siguientes tres funciones: `shouldSubscribeToParticipant`, `shouldPublishFromParticipant` y `stageStreamsToPublishForParticipant`. Todas se analizan a continuación.

Suscripción a participantes

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

Cuando un participante remoto se incorpora al escenario, el SDK consulta la aplicación host sobre el estado de suscripción deseado de ese participante. Las opciones son `NONE`, `AUDIO_ONLY` y `AUDIO_VIDEO`. Al devolver un valor para esta función, la aplicación host no tiene que preocuparse por el estado de la publicación, el estado actual de la suscripción ni el estado de la conexión del escenario. Si se devuelve `AUDIO_VIDEO`, el SDK espera a que el participante remoto publique antes de suscribirse y actualiza la aplicación host a través del renderizador durante todo el proceso.

Este es un ejemplo de implementación:

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

```
return Stage.SubscribeType.AUDIO_VIDEO;
}
```

Esta es la implementación completa de esta función para una aplicación host que siempre quiere que todos los participantes se vean entre sí; por ejemplo, una aplicación de videochat.

También son posibles implementaciones más avanzadas. Utilice la propiedad `userInfo` en `ParticipantInfo` para suscribirse de forma selectiva a los participantes en función de los atributos proporcionados por el servidor:

```
@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;
    }
}
```

Esto se puede utilizar para crear un escenario en el que los moderadores puedan monitorear a todos los invitados sin que ellos mismos los vean ni los escuchen. La aplicación host podría utilizar una lógica empresarial adicional para permitir que los moderados se vean entre sí, pero permanezcan invisibles para los invitados.

Configuración de la suscripción a participantes

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

Si se está suscribiendo a un participante remoto (consulte [Suscripción a participantes](#)), el SDK consulta a la aplicación host sobre una configuración de la suscripción personalizada para ese participante. Esta configuración es opcional y permite a la aplicación host controlar determinados aspectos del comportamiento de los suscriptores. Para obtener información sobre lo que se puede configurar, consulte [SubscribeConfiguration](#) en la documentación de referencia del SDK.

Este es un ejemplo de implementación:

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

    config.jitterBuffer.setMinDelay(JitterBufferConfiguration.JitterBufferDelay.MEDIUM());

    return config;
}
```

Esta implementación actualiza el retraso mínimo del búfer de fluctuación para todos los participantes suscritos al valor preestablecido MEDIUM.

Como en el caso de `shouldSubscribeToParticipant`, también son posibles implementaciones más avanzadas. El valor de `ParticipantInfo` proporcionado se puede utilizar para actualizar selectivamente la configuración de suscripción para participantes específicos.

Se recomienda utilizar los comportamientos predeterminados. Especifique la configuración personalizada solo si hay un comportamiento en particular que desee cambiar.

Publicación

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

Una vez realizada la conexión al escenario, el SDK consulta la aplicación host para ver si un participante en particular tiene que publicar. Esto solo se invoca en los participantes locales que tienen permiso para publicar en función del token proporcionado.

Este es un ejemplo de implementación:

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

Esto es para una aplicación de videochat estándar en la que los usuarios siempre quieren publicar. Pueden activar y desactivar el sonido y el video para ocultarse o verse y escucharse al instante. (También pueden usar la opción de publicar o anular la publicación, pero esto es mucho más lento.)

Es preferible silenciar o activar el sonido en casos de uso en los que se quiera cambiar la visibilidad de manera frecuente).

Elección de las transmisiones que publicar

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

Al publicar, esto se utiliza para determinar qué transmisiones de audio y video se tienen que publicar. Esto se explica en mayor detalle más adelante en [Publicación de una transmisión multimedia](#).

Actualización de la estrategia

La estrategia pretende ser dinámica: los valores devueltos por cualquiera de las funciones anteriores se pueden cambiar en cualquier momento. Por ejemplo, si la aplicación host no quiere publicar hasta que el usuario final presione un botón, puede devolver una variable de `shouldPublishFromParticipant` (como `hasUserTappedPublishButton`).

Cuando esa variable cambie en función de una interacción del usuario final, llame a `stage.refreshStrategy()` para indicar al SDK que debe consultar la estrategia a fin de obtener los valores más recientes y aplicar solo los cambios. Si el SDK observa que el valor `shouldPublishFromParticipant` cambió, iniciará el proceso de publicación. Si las consultas del SDK y todas las funciones devuelven el mismo valor que antes, la llamada a `refreshStrategy` no hará ninguna modificación en el escenario.

Si el valor devuelto de `shouldSubscribeToParticipant` cambia de `AUDIO_VIDEO` a `AUDIO_ONLY`, la transmisión de video se eliminará para todos los participantes con valores devueltos modificados, si ya existía con anterioridad una transmisión de video.

Por lo general, el escenario utiliza la estrategia para aplicar de la manera más eficiente la diferencia entre las estrategias anteriores y actuales, sin que la aplicación host tenga que preocuparse por todo el estado necesario para administrarla correctamente. Por eso, piense en la llamada a `stage.refreshStrategy()` como una operación barata, porque no hace nada a no ser que cambie la estrategia.

Renderer

La interfaz `StageRenderer` comunica el estado del escenario a la aplicación host. Por lo general, los eventos proporcionados por el renderizador permiten el funcionamiento completo de las

actualizaciones de la interfaz de usuario de la aplicación host. El renderizador ofrece el siguiente resultado:

```
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);
```

Para la mayoría de estos métodos, se proporcionan el Stage y la ParticipantInfo correspondientes.

No se espera que la información que proporciona el renderizador afecte a los valores de retorno de la estrategia. Por ejemplo, no se espera que el valor devuelto de `shouldSubscribeToParticipant` cambie cuando se llama a `onParticipantPublishStateChanged`. Si la aplicación host quiere suscribirse a un participante en particular, debe devolver el tipo de suscripción deseado, independientemente del estado de publicación de ese participante. El SDK es responsable de garantizar que se aplique el estado deseado de la estrategia en el momento correcto según el estado del escenario.

`StageRenderer` se puede adjuntar a la clase del escenario:

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

Tenga en cuenta que solo los participantes que publican desencadenan `onParticipantJoined`. `onParticipantLeft` se desencadena cada vez que un participante deja de publicar o abandona la sesión del escenario.

Publicación de una transmisión multimedia

Los dispositivos locales, como las cámaras y los micrófonos integrados, se detectan a través de `DeviceDiscovery`. A continuación, se muestra un ejemplo de cómo seleccionar la cámara frontal y el micrófono predeterminados y devolverlos como `LocalStageStreams` para que los publique el 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;
}
```

Visualización y eliminación de participantes

Cuando se complete la suscripción, recibirá una matriz de objetos `StageStream` a través de la función `onStreamsAdded` del renderizador. Puede obtener la vista previa de una `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);
```

Puede recuperar las estadísticas de audio de una `AudioStageStream`:

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

Cuando un participante deja de publicar o anula su suscripción, se invoca la función `onStreamsRemoved` con las transmisiones que se eliminaron. Las aplicaciones host tienen que usar esto como una señal para eliminar la transmisión de video del participante de la jerarquía de visualización.

`onStreamsRemoved` se invoca para todas las situaciones en las que se puede eliminar una transmisión, como las siguientes:

- El participante remoto deja de publicar.
- Un dispositivo local cancela la suscripción o cambia la suscripción de `AUDIO_VIDEO` a `AUDIO_ONLY`.
- El participante remoto abandona el escenario.
- El participante local abandona el escenario.

Ya que `onStreamsRemoved` se invoca en todas las situaciones, no es necesaria una lógica empresarial personalizada para eliminar a los participantes de la interfaz de usuario durante las operaciones de licencia remota o local.

Activación y desactivación del sonido de las transmisiones multimedia

Los objetos de `LocalStageStream` tienen una función `setMuted` que controla si la transmisión está silenciada. Esta función se puede invocar en la transmisión antes o después de que la devuelva la función de estrategia `streamsToPublishForParticipant`.

Importante: Si `streamsToPublishForParticipant` devuelve una nueva instancia de objeto de `LocalStageStream` después de una llamada a `refreshStrategy`, el estado de sonido desactivado del nuevo objeto de transmisión se aplicará al escenario. Tenga cuidado al crear nuevas instancias `LocalStageStream` para asegurarse de que se mantenga el estado de sonido desactivado que se espera.

Monitoreo del estado de sonido desactivado en los contenidos multimedia del participante remoto

Cuando un participante cambia el estado de sonido desactivado de su transmisión de video o audio, se invoca la función `onStreamMutedChanged` de renderizado con una lista de las transmisiones que cambiaron. Use el método `getMuted` en `StageStream` para actualizar la interfaz de usuario según corresponda.

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

Obtención de estadísticas de WebRTC

Para obtener las estadísticas más recientes de WebRTC de una transmisión de publicación o de suscripción, utilice `requestRTCStats` en `StageStream`. Cuando se complete una recopilación, recibirá estadísticas a través de `StageStream.Listener`, que se puede configurar en `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());
        }
    }
}
```

Obtención de los atributos de los participantes

Si especifica atributos en la solicitud del punto de conexión de `CreateParticipantToken`, puede ver los atributos en las propiedades de `ParticipantInfo`:

```
@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());
    }
}
```

Continuación de la sesión en segundo plano

Cuando la aplicación pase a segundo plano, es posible que quiera dejar de publicar o suscribirse solo al audio de otros participantes remotos. Para ello, actualice la implementación `Strategy` para detener la publicación y suscríbase a `AUDIO_ONLY` (o a `NONE`, si corresponde).

```
// 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();
}
```

Activación o desactivación de la codificación en capas con transmisión simultánea

Al publicar una transmisión multimedia, el SDK transmite transmisión de video de alta y baja calidad, de modo que los participantes remotos puedan suscribirse a transmisión incluso si tienen un ancho de banda de enlace descendente limitado. La codificación en capas con transmisión simultánea está activada de forma predeterminada. Puede desactivarla mediante la clase `StageVideoConfiguration.Simulcast`:

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

ImageLocalStageStream cameraStream = new ImageLocalStageStream(frontCamera, config);

// Other Stage implementation code
```

Limitaciones de la configuración de video

El SDK no permite forzar el uso del modo vertical ni del horizontal mediante `StageVideoConfiguration.setSize(BroadcastConfiguration.Vec2 size)`. En la orientación vertical, la dimensión más pequeña se utiliza como el ancho; en la orientación horizontal, como la altura. Esto significa que las dos siguientes llamadas a `setSize` tendrán el mismo efecto en la configuración de video:

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

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

Gestión de los problemas de red

Cuando se pierde la conexión de red del dispositivo local, el SDK se intenta volver a conectar internamente sin que el usuario lleve a cabo ninguna acción. En algunos casos, el SDK no funciona de manera correcta y es necesario que el usuario actúe. Hay dos errores principales relacionados con la pérdida de la conexión de red:

- Código de error 1400, mensaje: “Se ha perdido la conexión de pares debido a un error de red desconocido”
- Código de error 1300, mensaje: “Se han agotado los reintentos”

Si se recibe el primer error, pero no el segundo, el SDK sigue conectado al escenario e intentará restablecer sus conexiones automáticamente. Como medida de seguridad, puede hacer una llamada a `refreshStrategy` sin cambiar los valores devueltos por el método de estrategia para iniciar un intento de reconexión manual.

Si se recibe el segundo error, los intentos de reconexión del SDK fallaron y el dispositivo local ya no está conectado al escenario. En este caso, intente reincorporarse al escenario con una llamada a `join` después de que se restablezca la conexión de red.

En general, el que aparezcan errores después de incorporarse a un escenario correctamente indica que el SDK no pudo restablecer la conexión. Cree un nuevo objeto `Stage` e intente incorporarse cuando mejoren las condiciones de la red.

Uso de micrófonos Bluetooth

Para publicar con dispositivos de micrófono Bluetooth, debe iniciar una conexión SCO Bluetooth:

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

Problemas conocidos y soluciones alternativas del SDK de transmisión para Android de IVS | Transmisión en tiempo real

Este documento enumera problemas conocidos que puede experimentar al utilizar el SDK de transmisión para Android de la transmisión en tiempo real de Amazon IVS y sugiere posibles soluciones alternativas.

- Cuando un dispositivo Android se pone en reposo y se activa, es posible que la vista previa esté congelada.

Solución alternativa: cree y utilice un nuevo Stage.

- Cuando un participante se incorpora con un token que utiliza otro participante, la primera conexión se pierde sin que se produzca un error específico.

Solución alternativa: ninguna.

- Hay un problema poco frecuente en el que el editor publica, pero el estado de publicación que reciben los suscriptores es `inactive`.

Solución alternativa: pruebe a salir de la sesión y, a continuación, reincorporarse. Si el problema persiste, cree un nuevo token para el publicador.

- Durante una sesión del escenario, se puede producir un problema intermitente y poco frecuente de distorsión de audio, por lo general en llamadas de mayor duración.

Solución alternativa: el participante con audio distorsionado puede abandonar la sesión y volver a unirse a ella o anular la publicación y volver a publicar su audio para solucionar el problema.

- No se admiten micrófonos externos cuando se hacen publicaciones en un escenario.

Solución alternativa: no utilice un micrófono externo conectado por USB para publicar en un escenario.

- No se admite el uso de `createSystemCaptureSources` para publicar en un escenario con pantalla compartida.

Solución alternativa: administre la captura del sistema de forma manual mediante orígenes de entrada de imágenes personalizados y orígenes de entrada de audio personalizados.

- Cuando se elimina una `ImagePreviewView` de un elemento principal (por ejemplo, se llama a `removeView()` en el elemento principal), `ImagePreviewView` se libera inmediatamente. La `ImagePreviewView` no muestra ningún fotograma cuando se agrega a otra vista principal.

Solución alternativa: solicite otra vista previa mediante `getPreview`.

- Al incorporarse a un escenario con un Samsung Galaxy S22/+ con Android 12, es posible que aparezca un error 1401 y que el dispositivo local no pueda incorporarse al escenario o lo haga, pero no haya audio.

Solución alternativa: actualice a Android 13.

- Al incorporarse a un escenario con un Nokia X20 en Android 13, es posible que la cámara no se abra y se produzca una excepción.

Solución alternativa: ninguna.

- Es posible que los dispositivos con el conjunto de chips MediaTek Helio no reproduzcan correctamente el video de los participantes remotos.

Solución alternativa: ninguna.

- En algunos dispositivos, el sistema operativo del dispositivo puede elegir un micrófono diferente al seleccionado a través del SDK. Esto se debe a que el SDK de transmisión de Amazon IVS no puede controlar la forma en que se define la ruta de audio de `VOICE_COMMUNICATION`, ya que varía según los distintos fabricantes de dispositivos.

Solución alternativa: ninguna.

- Algunos codificadores de video de Android no se pueden configurar con un tamaño de video inferior a 176 x 176. La configuración de un tamaño más pequeño provoca un error e impide la transmisión.

Solución alternativa: no configure el tamaño del video para que sea inferior a 176 x 176.

Control de errores en el SDK de transmisión para Android de IVS | Transmisión en tiempo real

En esta sección se ofrece información general sobre las condiciones de error, cómo el SDK de transmisión para Android las notifica a la aplicación y qué debe hacer una aplicación cuando se producen esos errores.

Errores fatales frente a errores no fatales

El objeto de error tiene un campo booleano “es grave” de `BroadcastException`.

En general, los errores fatales están relacionados con la conexión al servidor de etapas (o bien no se puede establecer una conexión o bien se pierde y no se puede recuperar). La aplicación debe volver a crear el escenario y volver a unirse, posiblemente con un nuevo token o cuando se recupere la conectividad del dispositivo.

Los errores no fatales suelen estar relacionados con el estado de publicación/suscripción y son gestionados por el SDK, que reintenta la operación de publicación/suscripción.

Puede comprobar esta propiedad:

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

Errores de unión

Token con formato incorrecto

Esto ocurre cuando el token de la etapa tiene un formato incorrecto.

El SDK genera una excepción de Java al llamar a `stage.join`, con el código de error =1000 y `fatal=true`.

Acción: cree un token válido e intente unirse de nuevo.

Token vencido

Esto ocurre cuando el token de la etapa está caducado.

El SDK genera una excepción de Java al llamar a `stage.join`, con el código de error =1001 y `fatal=true`.

Acción: cree un token nuevo e intente unirse de nuevo.

Token no válido o revocado

Esto ocurre cuando el token de la etapa no tiene un formato incorrecto, sino que el servidor de etapas lo rechaza. Esto se informa de forma asíncrona a través del renderizador de la etapa suministrado por la aplicación.

El SDK llama a `onConnectionStateChanged` con una excepción, con el código de error =1026 y `fatal =true`.

Acción: cree un token válido e intente unirse de nuevo.

Errores de red durante la unión inicial

Esto ocurre cuando el SDK no puede ponerse en contacto con el servidor de etapas para establecer una conexión. Esto se informa de forma asíncrona a través del renderizador de la etapa suministrado por la aplicación.

El SDK llama a `onConnectionStateChanged` con una excepción, con el código de error =1300 y `fatal =true`.

Acción: espere a que se recupere la conectividad del dispositivo e intente conectarse de nuevo.

Errores de red cuando ya está conectado

Si se interrumpe la conexión de red del dispositivo, es posible que el SDK pierda la conexión con los servidores de etapas. Esto se informa de forma asíncrona a través del renderizador de la etapa suministrado por la aplicación.

El SDK llama a `onConnectionStateChanged` con una excepción, con el código de error =1300 y `fatal =true`.

Acción: espere a que se recupere la conectividad del dispositivo e intente conectarse de nuevo.

Errores de publicación/suscripción

Inicial

Existen varios errores:

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

Estos se informan de forma asíncrona a través del renderizador de la etapa suministrado por la aplicación.

El SDK vuelve a intentar la operación un número limitado de veces. Durante los reintentos, el estado de publicación/suscripción es `ATTEMPTING_PUBLISH/ATTEMPTING_SUBSCRIBE`. Si el reintento se realiza correctamente, el estado cambia a `PUBLISHED/SUBSCRIBED`.

El SDK llama a `onError` con el código de error correspondiente y `fatal = false`.

Acción: no es necesario realizar ninguna acción, ya que el SDK vuelve a intentarlo automáticamente. Si lo desea, la aplicación puede actualizar la estrategia para forzar más reintentos.

Ya está establecido, luego falla

Una publicación o una suscripción pueden fallar una vez establecidas, muy probablemente debido a un error de red. El código de error para “Se ha perdido la conexión de pares debido a un error de red” es 1400.

Esto se informa de forma asíncrona a través del renderizador de la etapa suministrado por la aplicación.

El SDK vuelve a intentar la operación de publicación/suscripción. Durante los reintentos, el estado de publicación/suscripción es `ATTEMPTING_PUBLISH/ATTEMPTING_SUBSCRIBE`. Si el reintento se realiza correctamente, el estado cambia a `PUBLISHED/SUBSCRIBED`.

El SDK llama a `onError` con el código de error =1400 y `fatal = false`.

Acción: no es necesario realizar ninguna acción, ya que el SDK vuelve a intentarlo automáticamente. Si lo desea, la aplicación puede actualizar la estrategia para forzar más reintentos. En caso de pérdida total de conectividad, es probable que la conexión a las etapas también falle.

SDK de transmisión de IVS: guía para iOS | Transmisión en tiempo real

El SDK de transmisión para transmisión en tiempo real de IVS para iOS permite a los participantes enviar y recibir videos en iOS.

El módulo `AmazonIVSBroadcast` implementa la interfaz descrita en este documento. Se admiten las siguientes operaciones:

- Incorporación a un escenario

- Publicación de contenido multimedia para otros participantes del escenario
- Suscripción al contenido multimedia de otros participantes del escenario
- Administración y monitoreo del video y audio publicados en el escenario
- Obtención de estadísticas de WebRTC de cada conexión de pares
- Todas las operaciones del SDK de transmisión para iOS de transmisión de baja latencia de IVS

Versión más reciente del SDK de transmisión para iOS: 1.23.0 ([notas de la versión](#))

Documentación de referencia: a fin de obtener información sobre los métodos más importantes disponibles en el SDK de transmisión de iOS de Amazon IVS, consulte la documentación de referencia en <https://aws.github.io/amazon-ivs-broadcast-docs/1.23.0/ios/>.

Código de muestra: consulte el repositorio de muestra de iOS en GitHub: <https://github.com/aws-samples/amazon-ivs-broadcast-ios-sample>.

Requisitos de la plataforma: iOS 14 o posterior

Introducción al SDK de transmisión para iOS de IVS | Transmisión en tiempo real

Este documento explica los pasos necesarios para comenzar a utilizar el SDK de transmisión para iOS de la transmisión en tiempo real de IVS.

Instalación de la biblioteca

Recomendamos que integre el SDK de transmisión a través de CocoaPods. (También puede agregar el marco a su proyecto de forma manual).

Recomendación: integrar el SDK de la transmisión (CocoaPods)

La funcionalidad en tiempo real se publica como una subespecificación del SDK de transmisión para transmisión de baja latencia de iOS. De este modo, los clientes pueden elegir incluirla o excluirla en función de sus necesidades. Incluirla aumenta el tamaño del paquete.

Las versiones se publican a través de CocoaPods bajo el nombre AmazonIVSBroadcast. Agregue esta dependencia a su Podfile:

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

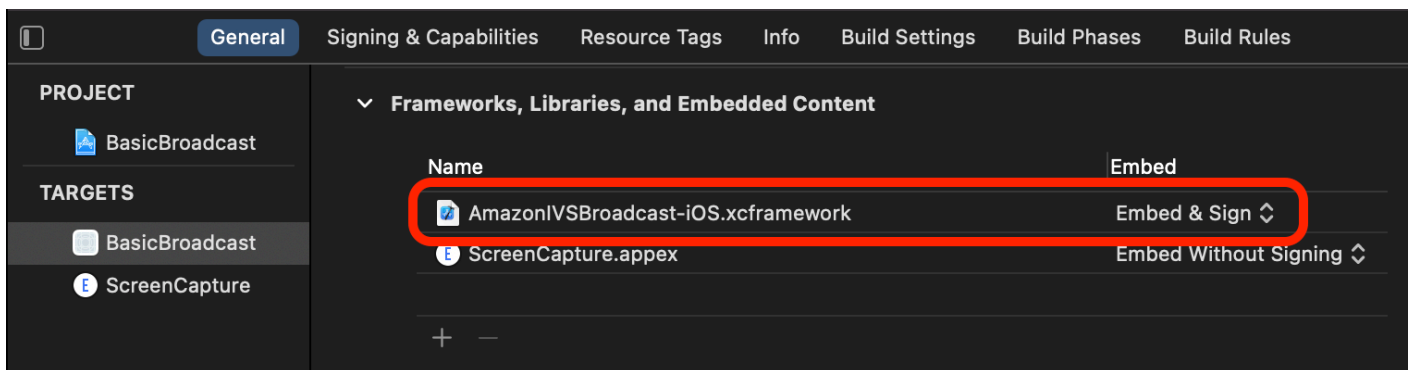
Ejecute `pod install` y el SDK estará disponible en su `.xcworkspace`.

Importante: El SDK de transmisión para transmisión en tiempo real de IVS (es decir, con la subespecificación de escenario) incluye todas las características del SDK de transmisión para transmisión de baja latencia de IVS. No es posible integrar ambos SDK en el mismo proyecto. Si agrega a su proyecto la subespecificación de escenarios mediante CocoaPods, asegúrese de eliminar cualquier otra línea del Podfile que contenga `AmazonIVSBroadcast`. Por ejemplo, no incluya estas dos líneas en el Podfile:

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

Método alternativo: instalar el marco de forma manual

1. Descargue la versión más reciente desde <https://broadcast.live-video.net/1.23.0/AmazonIVSBroadcast-Stages.xcframework.zip>.
2. Extraiga el contenido del archivo. `AmazonIVSBroadcast.xcframework` contiene el SDK para el dispositivo y el simulador.
3. Integre el `AmazonIVSBroadcast.xcframework` arrastrándolo a la sección Frameworks, Libraries, and Embedded Content (Marcos, bibliotecas y contenido integrado) de la pestaña General para el destino de la aplicación.



Solicitar permisos

La aplicación debe solicitar permiso para acceder a la cámara y al micrófono del usuario. (Esto no es específico de Amazon IVS; es necesario para cualquier aplicación que necesite acceso a cámaras y micrófonos).

Aquí, verificamos si el usuario ya ha concedido permisos y, de no ser así, preguntamos por ellos:

```
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.
}
```

Es necesario hacer esto para los tipos de medios `.video` y `.audio`, si desea tener acceso a cámaras y micrófonos, respectivamente.

También es necesario agregar entradas para `NSCameraUsageDescription` y `NSMicrophoneUsageDescription` para su `Info.plist`. De lo contrario, la aplicación se bloqueará al intentar solicitar permisos.

Desactivar el temporizador inactivo de la aplicación

Esto es opcional, pero recomendable. Evita que el dispositivo se ponga en suspensión mientras utiliza el SDK de transmisión, lo que interrumpiría la transmisión.

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

Publicación y suscripción con el SDK de transmisión para iOS de IVS | Transmisión en tiempo real

Este documento explica los pasos necesarios para publicar y suscribirse a una fase mediante el SDK de transmisión para iOS de la transmisión en tiempo real de IVS.

Conceptos

Los siguientes tres conceptos básicos subyacen a la funcionalidad de transmisión en tiempo real: [escenario](#), [estrategia](#) y [renderizador](#). El objetivo del diseño es minimizar la cantidad de lógica necesaria por parte del cliente para crear un producto que funcione.

Escenario

La clase `IVSStage` es el principal punto de interacción entre la aplicación host y el SDK. La clase representa el escenario como tal y se usa para entrar y salir de él. Para crear un escenario o incorporarse a él, es necesaria una cadena de símbolos válida y que no haya vencido del plano de control (representada como token). Entrar a un escenario y salir de él es sencillo.

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

try stage.join()

stage.leave()
```

También se pueden adjuntar `IVSStageRenderer` y `IVSErrorDelegate` en la clase `IVSStage`:

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

Strategy (Estrategia)

El protocolo `IVSStageStrategy` proporciona una forma para que la aplicación host comunique el estado deseado del escenario al SDK. Es necesario implementar las siguientes tres funciones: `shouldSubscribeToParticipant`, `shouldPublishParticipant` y `streamsToPublishForParticipant`. Todas se analizan a continuación.

Suscripción a participantes

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

Cuando un participante remoto se une a un escenario, el SDK consulta a la aplicación host sobre el estado de suscripción deseado para ese participante. Las opciones son `.none`, `.audioOnly` y

.audioVideo. Al devolver un valor para esta función, la aplicación host no tiene que preocuparse por el estado de la publicación, el estado actual de la suscripción ni el estado de la conexión del escenario. Si se devuelve .audioVideo, el SDK espera a que el participante remoto publique antes de suscribirse y actualiza la aplicación host a través del renderizador durante todo el proceso.

Este es un ejemplo de implementación:

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

Esta es la implementación completa de esta función para una aplicación host que siempre quiere que todos los participantes se vean entre sí; por ejemplo, una aplicación de videochat.

También son posibles implementaciones más avanzadas. Utilice la propiedad `attributes` en `IVSParticipantInfo` para suscribirse de forma selectiva a los participantes en función de los atributos proporcionados por el servidor:

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

Esto se puede utilizar para crear un escenario en el que los moderadores puedan monitorear a todos los invitados sin que ellos mismos los vean ni los escuchen. La aplicación host podría utilizar una lógica empresarial adicional para permitir que los moderadores se vean entre sí, pero permanezcan invisibles para los invitados.

Configuración de la suscripción a participantes

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

Si se está suscribiendo a un participante remoto (consulte [Suscripción a participantes](#)), el SDK consulta a la aplicación host sobre una configuración de la suscripción personalizada para ese

participante. Esta configuración es opcional y permite a la aplicación host controlar determinados aspectos del comportamiento de los suscriptores. Para obtener información sobre lo que se puede configurar, consulte [SubscribeConfiguration](#) en la documentación de referencia del SDK.

Este es un ejemplo de implementación:

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

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

  return config
}
```

Esta implementación actualiza el retraso mínimo del búfer de fluctuación para todos los participantes suscritos al valor preestablecido MEDIUM.

Como en el caso de `shouldSubscribeToParticipant`, también son posibles implementaciones más avanzadas. El valor de `ParticipantInfo` proporcionado se puede utilizar para actualizar selectivamente la configuración de suscripción para participantes específicos.

Se recomienda utilizar los comportamientos predeterminados. Especifique la configuración personalizada solo si hay un comportamiento en particular que desee cambiar.

Publicación

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

Una vez realizada la conexión al escenario, el SDK consulta la aplicación host para ver si un participante en particular tiene que publicar. Esto solo se invoca en los participantes locales que tienen permiso para publicar en función del token proporcionado.

Este es un ejemplo de implementación:

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


Esto es para una aplicación de videochat estándar en la que los usuarios siempre quieren publicar. Pueden activar y desactivar el sonido y el video para ocultarse o verse y escucharse al instante. (También pueden usar la opción de publicar o anular la publicación, pero esto es mucho más lento. Es preferible silenciar o activar el sonido en casos de uso en los que se quiera cambiar la visibilidad de manera frecuente).

Elección de las transmisiones que publicar

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

Al publicar, esto se utiliza para determinar qué transmisiones de audio y video se tienen que publicar. Esto se explica en mayor detalle más adelante en [Publicación de una transmisión multimedia](#).

Actualización de la estrategia

La estrategia pretende ser dinámica: los valores devueltos por cualquiera de las funciones anteriores se pueden cambiar en cualquier momento. Por ejemplo, si la aplicación host no quiere publicar hasta que el usuario final presione un botón, puede devolver una variable de `shouldPublishParticipant` (como `hasUserTappedPublishButton`). Cuando esa variable cambie en función de una interacción del usuario final, llame a `stage.refreshStrategy()` para indicar al SDK que debe consultar la estrategia a fin de obtener los valores más recientes y aplicar solo los cambios. Si el SDK observa que el valor `shouldPublishParticipant` cambió, iniciará el proceso de publicación. Si las consultas del SDK y todas las funciones devuelven el mismo valor que antes, la llamada `refreshStrategy` no hará ninguna modificación en el escenario.

Si el valor devuelto de `shouldSubscribeToParticipant` cambia de `.audioVideo` a `.audioOnly`, la transmisión de video se eliminará para todos los participantes con valores devueltos modificados, si ya existía con anterioridad una transmisión de video.

Por lo general, el escenario utiliza la estrategia para aplicar de la manera más eficiente la diferencia entre las estrategias anteriores y actuales, sin que la aplicación host tenga que preocuparse por todo el estado necesario para administrarla correctamente. Por eso, piense en la llamada a `stage.refreshStrategy()` como una operación barata, porque no hace nada a no ser que cambie la estrategia.

Renderer

El protocolo `IVSStageRenderer` comunica el estado del escenario a la aplicación host. Por lo general, los eventos proporcionados por el renderizador permiten el funcionamiento completo de las

actualizaciones de la interfaz de usuario de la aplicación host. El renderizador ofrece el siguiente resultado:

```
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?)
```

No se espera que la información que proporciona el renderizador afecte a los valores de retorno de la estrategia. Por ejemplo, no se espera que el valor devuelto de `shouldSubscribeToParticipant` cambie cuando se llama a `participant:didChangePublishState`. Si la aplicación host quiere suscribirse a un participante en particular, debe devolver el tipo de suscripción deseado, independientemente del estado de publicación de ese participante. El SDK es responsable de garantizar que se aplique el estado deseado de la estrategia en el momento correcto según el estado del escenario.

Tenga en cuenta que solo los participantes que publican desencadenan `participantDidJoin`. `participantDidLeave` se desencadena cada vez que un participante deja de publicar o abandona la sesión del escenario.

Publicación de una transmisión multimedia

Los dispositivos locales, como las cámaras y los micrófonos integrados, se detectan a través de `IVSDeviceDiscovery`. A continuación, se muestra un ejemplo de cómo seleccionar la cámara

frontal y el micrófono predeterminados y devolverlos como `IVSLocalStageStreams` para que los publique el 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]
}
```

Visualización y eliminación de participantes

Cuando se complete la suscripción, recibirá una matriz de objetos `IVSStageStream` a través de la función `didAddStreams` del renderizador. Para obtener una vista previa o recibir estadísticas del audio de este participante, puede acceder al objeto `IVSDevice` subyacente desde la transmisión:

```
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 */
    })
}
```

Cuando un participante deja de publicar o anula su suscripción, se invoca la función `didRemoveStreams` con las transmisiones que se eliminaron. Las aplicaciones host tienen que usar esto como una señal para eliminar la transmisión de video del participante de la jerarquía de visualización.

`didRemoveStreams` se invoca para todas las situaciones en las que se puede eliminar una transmisión, como las siguientes:

- El participante remoto deja de publicar.
- Un dispositivo local cancela la suscripción o cambia la suscripción de `.audioVideo` a `.audioOnly`.
- El participante remoto abandona el escenario.
- El participante local abandona el escenario.

Ya que `didRemoveStreams` se invoca en todas las situaciones, no es necesaria una lógica empresarial personalizada para eliminar a los participantes de la interfaz de usuario durante las operaciones de licencia remota o local.

Activación y desactivación del sonido de las transmisiones multimedia

Los objetos de `IVSLocalStageStream` tienen una función `setMuted` que controla si la transmisión está silenciada. Esta función se puede invocar en la transmisión antes o después de que la devuelva la función de estrategia `streamsToPublishForParticipant`.

Importante: Si `streamsToPublishForParticipant` devuelve una nueva instancia de objeto de `IVSLocalStageStream` después de una llamada a `refreshStrategy`, el estado de sonido desactivado del nuevo objeto de transmisión se aplicará al escenario. Tenga cuidado al crear nuevas instancias `IVSLocalStageStream` para asegurarse de que se mantenga el estado de sonido desactivado que se espera.

Monitoreo del estado de sonido desactivado en los contenidos multimedia del participante remoto

Cuando un participante cambia el estado de sonido desactivado de su transmisión de video o audio, se invoca la función `didChangeMutedStreams` de renderizado con una lista de las transmisiones que cambiaron. Use la propiedad `isMuted` en `IVSStageStream` para actualizar su interfaz de usuario según corresponda:

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

Creación de una configuración de escenario

Para personalizar los valores de la configuración de video de un escenario, utilice `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
```

Obtención de estadísticas de WebRTC

Para obtener las estadísticas más recientes de WebRTC de una transmisión de publicación o de suscripción, utilice `requestRTCStats` en `IVSStageStream`. Cuando se complete una recopilación, recibirá estadísticas a través de `IVSStageStreamDelegate`, que se puede configurar en `IVSStageStream`. Para recopilar continuamente estadísticas de WebRTC, llame a esta función en un `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)")
        }
    }
}
```

Obtención de los atributos de los participantes

Si especifica atributos en la solicitud del punto de conexión de `CreateParticipantToken`, puede ver los atributos en las propiedades de `IVSParticipantInfo`:

```
func stage(_ stage: IVSStage, participantDidJoin participant: IVSParticipantInfo) {
    print("ID: \(participant.participantId)")
    for attribute in participant.attributes {
        print("attribute: \(attribute.key)=\(attribute.value)")
    }
}
```

Continuación de la sesión en segundo plano

Cuando la aplicación pasa a segundo plano, puede seguir en el escenario mientras escucha el audio remoto, aunque no podrá enviar su propia imagen y audio. Es necesario que actualice la implementación `IVSStrategy` para detener la publicación y se suscriba a `.audioOnly` (o a `.none`, si corresponde):

```
func stage(_ stage: IVSStage, shouldPublishParticipant participant: IVSParticipantInfo)
    -> Bool {
    return false
}
func stage(_ stage: IVSStage, shouldSubscribeToParticipant participant:
    IVSParticipantInfo) -> IVSStageSubscribeType {
    return .audioOnly
}
```

A continuación, haga una llamada a `stage.refreshStrategy()`.

Activación o desactivación de la codificación en capas con transmisión simultánea

Al publicar una transmisión multimedia, el SDK transmite transmisión de video de alta y baja calidad, de modo que los participantes remotos puedan suscribirse a transmisión incluso si tienen un ancho de banda de enlace descendente limitado. La codificación en capas con transmisión simultánea está activada de forma predeterminada. Puede desactivarla con `IVSSimulcastConfiguration`:

```
// Disable Simulcast
let config = IVSLocalStageStreamVideoConfiguration()
config.simulcast.enabled = false

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

// Other Stage implementation code
```

Transmisión del escenario a un canal de IVS

Para transmitir un escenario, cree una `IVSBroadcastSession` independiente y, a continuación, siga las instrucciones habituales para la transmisión con el SDK, descritas con anterioridad. La propiedad `device` en `IVSStageStream` será un `IVSImageDevice` o `IVSAudioDevice`, tal y como se muestra en el fragmento anterior. Estos se pueden conectar a `IVSBroadcastSession.mixer` para transmitir todo el escenario en un diseño personalizable.

Si lo desea, puede componer un escenario y transmitirlo a un canal de IVS de baja latencia para llegar a un público más amplio. Consulte [Habilitación de varios hosts en una transmisión de Amazon IVS](#) en la Guía del usuario de transmisión de baja latencia.

Cómo elige iOS la resolución de la cámara y la velocidad de fotogramas

La cámara administrada por el SDK de transmisión optimiza su resolución y velocidad de fotogramas (fotogramas por segundo o FPS) para minimizar la producción de calor y el consumo de energía. En esta sección se explica de qué manera se seleccionan la resolución y la velocidad de fotogramas para ayudar a las aplicaciones de host a optimizar los casos de uso.

Al crear una `IVSLocalStageStream` a una `IVSCamera`, la cámara se optimiza para una velocidad de fotogramas `IVSLocalStageStreamVideoConfiguration.targetFramerate` y una resolución de `IVSLocalStageStreamVideoConfiguration.size`. Al llamar a `IVSLocalStageStream.setConfiguration` se actualiza la cámara con valores más recientes.

Vista previa de cámara

Si crea una vista previa de una `IVSCamera` sin adjuntarla a una `IVSBroadcastSession` o `IVSStage`, la resolución predeterminada será de 1080p y la velocidad de fotogramas de 60 fps.

Transmisión de un escenario

Cuando se utiliza una `IVSBroadcastSession` para transmitir un `IVSStage`, el SDK intenta optimizar la cámara con una resolución y una velocidad de fotogramas que cumplan con los criterios de ambas sesiones.

Por ejemplo, si la configuración de transmisión está configurada para tener una velocidad de fotogramas de 15 FPS y una resolución de 1080p, mientras que el escenario tiene una velocidad de fotogramas de 30 FPS y una resolución de 720p, el SDK seleccionará una configuración de cámara con una velocidad de fotogramas de 30 FPS y una resolución de 1080p.

La `IVSBroadcastSession` eliminará el resto de fotogramas de la cámara y `IVSStage` escalará la imagen de 1080p a 720p.

Si una aplicación host planea usar tanto `IVSBroadcastSession` como `IVSStage`, con una cámara, recomendamos que las propiedades de `targetFramerate` y `size` de las configuraciones respectivas coincidan. Si no coinciden, la cámara podría reconfigurarse a sí misma al capturar video, lo que provocaría un breve retraso en la entrega de las muestras de video.

Si tener valores idénticos no cumple con los requisitos de uso de la aplicación host, crear primero la cámara de mayor calidad evitará que la cámara se reconfigure por sí sola cuando se agregue la sesión de menor calidad. Por ejemplo, si emite a 1080p y 30 FPS y, posteriormente, se une a un escenario configurado en 720p y 30 FPS, la cámara no se reconfigurará por sí sola y el video continuará sin interrupciones. Esto se debe a que 720p es inferior o igual a 1080p y 30 FPS es inferior o igual a 30 FPS.

Velocidades de fotogramas, resoluciones y proporciones de aspecto arbitrarias

La mayoría del hardware de las cámaras puede coincidir exactamente con los formatos habituales, como 720p a 30 FPS o 1080p a 60 FPS. Sin embargo, no se puede hacer coincidir exactamente con todos los formatos. El SDK de transmisión elige la configuración de la cámara según las siguientes reglas (en orden de prioridad):

1. El ancho y el alto de la resolución son mayores o iguales a la resolución deseada, pero dentro de esta restricción, el ancho y el alto son lo más pequeños posible.
2. La velocidad de fotogramas es mayor o igual a la velocidad de fotogramas deseada, pero dentro de esta restricción, la velocidad de fotogramas es lo más baja posible.
3. La relación de aspecto coincide con la relación de aspecto deseada.
4. Si hay varios formatos coincidentes, se utiliza el formato con el mayor campo de visión.

A continuación, se incluyen dos ejemplos:

- La aplicación host está intentando transmitir en 4K a 120 FPS. La cámara seleccionada solo admite 4K a 60 FPS o 1080p a 120 FPS. El formato seleccionado será 4K a 60 FPS, porque la regla de resolución tiene mayor prioridad que la regla de velocidad de fotogramas.
- Se solicita una resolución irregular, 1910x1070. La cámara utilizará 1920x1080. Tenga cuidado: si elige una resolución como 1921x1080, la cámara escalará hasta la siguiente resolución disponible (por ejemplo, 2592x1944), lo que supone una penalización del ancho de banda de la memoria y de la CPU.

¿Qué pasa con Android?

Android no ajusta la resolución ni la velocidad de fotogramas sobre la marcha como lo hace iOS, por lo que esto no afecta al SDK de transmisión de Android.

Problemas conocidos y soluciones alternativas del SDK de transmisión para iOS de IVS | Transmisión en tiempo real

Este documento enumera problemas conocidos que puede experimentar al utilizar el SDK de transmisión para iOS de la transmisión en tiempo real de Amazon IVS y sugiere posibles soluciones alternativas.

- Cambiar las rutas de audio Bluetooth puede ser impredecible. Si conecta un dispositivo nuevo a mitad de la sesión, iOS puede o no cambiar automáticamente la ruta de entrada. Además, no es posible elegir entre varios auriculares Bluetooth conectados al mismo tiempo. Esto ocurre tanto en las sesiones normales de transmisión como en las del escenario.

Solución alternativa: si planea utilizar auriculares Bluetooth, conéctelos antes de iniciar la transmisión o el escenario y déjelos conectados durante toda la sesión.

- Los participantes que utilicen un iPhone 14, iPhone 14 Plus, iPhone 14 Pro o iPhone 14 Pro Max pueden provocar problemas de eco en otros participantes.

Solución alternativa: los participantes que utilicen los dispositivos afectados pueden utilizar auriculares para evitar que otros participantes sufran el problema de eco.

- Cuando un participante se incorpora con un token que utiliza otro participante, la primera conexión se pierde sin que se produzca un error específico.

Solución alternativa: ninguna.

- Hay un problema poco frecuente en el que el editor publica, pero el estado de publicación que reciben los suscriptores es `inactive`.

Solución alternativa: pruebe a salir de la sesión y, a continuación, reincorporarse. Si el problema persiste, cree un nuevo token para el publicador.

- Cuando un participante publica o se suscribe, es posible recibir un error con el código 1400 que indica la desconexión debido a un problema de red, incluso cuando la red es estable.

Solución alternativa: intente volver a publicar o a suscribirse.

- Durante una sesión del escenario, se puede producir un problema intermitente y poco frecuente de distorsión de audio, por lo general en llamadas de mayor duración.

Solución alternativa: el participante con audio distorsionado puede abandonar la sesión y volver a unirse a ella o anular la publicación y volver a publicar su audio para solucionar el problema.

Control de errores en el SDK de transmisión para iOS de IVS | Transmisión en tiempo real

En esta sección se ofrece información general sobre las condiciones de error, cómo el SDK de transmisión para iOS las notifica a la aplicación y qué debe hacer una aplicación cuando se producen esos errores.

Errores fatales frente a errores no fatales

El objeto de error tiene un booleano “es grave”. Se trata de una entrada de diccionario `IVSBroadcastErrorIsFatalKey` que contiene un booleano.

En general, los errores fatales están relacionados con la conexión al servidor de etapas (o bien no se puede establecer una conexión o bien se pierde y no se puede recuperar). La aplicación debe volver a crear el escenario y volver a unirse, posiblemente con un nuevo token o cuando se recupere la conectividad del dispositivo.

Los errores no fatales suelen estar relacionados con el estado de publicación/suscripción y son gestionados por el SDK, que reintenta la operación de publicación/suscripción.

Puede comprobar esta propiedad:

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

Errores de unión

Token con formato incorrecto

Esto ocurre cuando el token de la etapa tiene un formato incorrecto.

El SDK lanza una excepción de Swift con el código de error =1000 y `IVSBroadcastErrorisFatalKey =YES`.

Acción: cree un token válido e intente unirse de nuevo.

Token vencido

Esto ocurre cuando el token de la etapa está caducado.

El SDK lanza una excepción de Swift con el código de error =1001 y `IVSBroadcastErrorisFatalKey =YES`.

Acción: cree un token nuevo e intente unirse de nuevo.

Token no válido o revocado

Esto ocurre cuando el token de la etapa no tiene un formato incorrecto, sino que el servidor de etapas lo rechaza. Esto se informa de forma asíncrona a través del renderizador de la etapa suministrado por la aplicación.

El SDK llama a `stage(didChange connectionState, withError error)` con el código de error =1026 y `IVSBroadcastErrorisFatalKey =YES`.

Acción: cree un token válido e intente unirse de nuevo.

Errores de red durante la unión inicial

Esto ocurre cuando el SDK no puede ponerse en contacto con el servidor de etapas para establecer una conexión. Esto se informa de forma asíncrona a través del renderizador de la etapa suministrado por la aplicación.

El SDK llama a `stage(didChange connectionState, withError error)` con el código de error =1300 y `IVSBroadcastErrorisFatalKey =YES`.

Acción: espere a que se recupere la conectividad del dispositivo e intente conectarse de nuevo.

Errores de red cuando ya está conectado

Si se interrumpe la conexión de red del dispositivo, es posible que el SDK pierda la conexión con los servidores de etapas. Esto se informa de forma asíncrona a través del renderizador de la etapa suministrado por la aplicación.

El SDK llama `stage(didChange connectionState, withError error)` con el código de error =1300 y el valor de `IVSBroadcastErrorIsFatalKey =YES`.

Acción: espere a que se recupere la conectividad del dispositivo e intente conectarse de nuevo.

Errores de publicación/suscripción

Inicial

Existen varios errores:

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

Estos se informan de forma asíncrona a través del renderizador de la etapa suministrado por la aplicación.

El SDK vuelve a intentar la operación un número limitado de veces. Durante los reintentos, el estado de publicación/suscripción es `ATTEMPTING_PUBLISH/ATTEMPTING_SUBSCRIBE`. Si el reintento se realiza correctamente, el estado cambia a `PUBLISHED/SUBSCRIBED`.

El SDK llama a `IVSErrorDelegate:didEmitError` con el código de error correspondiente y `IVSBroadcastErrorIsFatalKey == NO`.

Acción: no es necesario realizar ninguna acción, ya que el SDK vuelve a intentarlo automáticamente. Si lo desea, la aplicación puede actualizar la estrategia para forzar más reintentos.

Ya está establecido, luego falla

Una publicación o una suscripción pueden fallar una vez establecidas, muy probablemente debido a un error de red. El código de error para “Se ha perdido la conexión de pares debido a un error de red” es 1400.

Esto se informa de forma asíncrona a través del renderizador de la etapa suministrado por la aplicación.

El SDK vuelve a intentar la operación de publicación/suscripción. Durante los reintentos, el estado de publicación/suscripción es `ATTEMPTING_PUBLISH/ATTEMPTING_SUBSCRIBE`. Si el reintento se realiza correctamente, el estado cambia a `PUBLISHED/SUBSCRIBED`.

El SDK llama a `didEmitError` con el código de error =1400 y `IVSBroadcastErrorIsFatalKey =NO`.

Acción: no es necesario realizar ninguna acción, ya que el SDK vuelve a intentarlo automáticamente. Si lo desea, la aplicación puede actualizar la estrategia para forzar más reintentos. En caso de pérdida total de conectividad, es probable que la conexión a las etapas también falle.

SDK de transmisión de IVS: orígenes de imágenes personalizados | Transmisión en tiempo real

Los orígenes de entrada de imágenes personalizados permiten que una aplicación proporcione su propia entrada de imagen al SDK de transmisión, en lugar de limitarse a las cámaras predeterminadas. Un origen de imagen personalizado puede ser tan simple como una marca de agua semitransparente o una escena estática de “vuelvo enseguida”, o puede permitir que la aplicación haga un procesamiento personalizado adicional, como agregar filtros de belleza a la cámara.

Cuando utiliza una fuente de entrada de imagen personalizada para el control personalizado de la cámara (como el uso de bibliotecas de filtros de belleza que requieren acceso a la cámara), el SDK de transmisión ya no es responsable de administrar la cámara. En cambio, la aplicación es responsable de manejar correctamente el ciclo de vida de la cámara. Consulte la documentación oficial de la plataforma sobre cómo su aplicación debe administrar la cámara.

Android

Después de crear una sesión de `DeviceDiscovery`, cree un origen de entrada de imagen:

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

Este método devuelve un `CustomImageSource`, que es una fuente de imagen respaldada por una Android [Surface](#) (Superficie) estándar. La subclase `SurfaceSource` se puede cambiar de tamaño y rotar. También puede crear un `ImagePreviewView` para mostrar una vista previa de su contenido.

Para recuperar el Surface subyacente :

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

Este Surface se puede usar como búfer de salida para productores de imágenes como Camera2, OpenGL ES y otras bibliotecas. El caso de uso más simple es dibujar directamente un mapa de bits estático o un color en el lienzo de la superficie. Sin embargo, muchas bibliotecas (como las bibliotecas de filtros de belleza) proporcionan un método que permite que una aplicación especifique un Surface externo para la representación. Puede usar dicho método para pasar el Surface a la biblioteca de filtros, lo que permite que la biblioteca genere fotogramas procesados para que la sesión de transmisión los transmita.

Este CustomImageSource se puede encapsular en una LocalStageStream. La StageStrategy lo puede devolver para publicarlo en un Stage.

iOS

Después de crear una sesión de DeviceDiscovery, cree un origen de entrada de imagen:

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

Este método devuelve un IVSCustomImageSource, que es una fuente de imagen que permite que la aplicación envíe CMSampleBuffers manualmente. Para conocer los formatos de píxeles admitidos, consulte la referencia del SDK de transmisión de iOS; un enlace a la versión más actual se encuentra en las [Notas de la versión de Amazon IVS](#) para la última versión del SDK de transmisión.

Las muestras enviadas al origen personalizado se transmitirán en el escenario:

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

Para transmitir video, utilice este método en una devolución de llamada. Por ejemplo, si está utilizando la cámara, cada vez que se recibe un nuevo búfer de muestra de un AVCaptureSession, la aplicación puede reenviar el búfer de muestra a la fuente de imagen personalizada. Si lo desea, la aplicación puede aplicar más procesamiento (como un filtro de belleza) antes de enviar la muestra a la fuente de imagen personalizada.

El IVSCustomImageSource se puede encapsular en IVSLocalStageStream. La IVSStageStrategy lo puede devolver para publicarlo en un Stage.

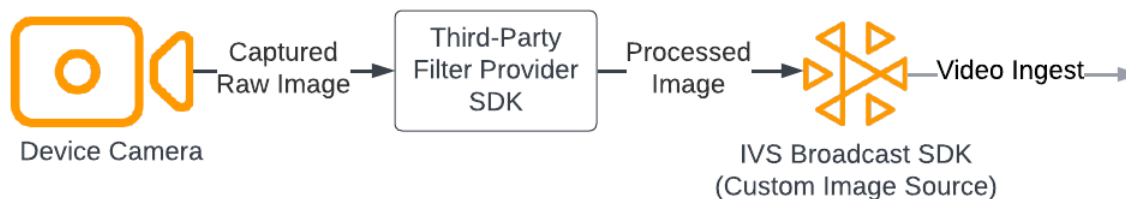
SDK de transmisión de IVS: filtros de cámara externos | Transmisión en tiempo real

Esta guía supone que ya está familiarizado con las fuentes de [imágenes personalizadas](#) y que ha integrado el [SDK de transmisión en tiempo real de IVS](#) en su aplicación.

Los filtros de cámara permiten a los creadores de transmisiones en directo aumentar o modificar su aspecto facial o de fondo. Esto puede aumentar la participación de los espectadores, atraerlos y mejorar la experiencia de transmisión en directo.

Integración de filtros de cámara externos

Puede integrar los SDK de filtros de cámara externos con el SDK de transmisión de IVS introduciendo la salida del SDK de filtros en una [fuente de entrada de imágenes personalizada](#). Una fuente de entrada de imágenes personalizada permite que una aplicación proporcione su propia entrada de imagen al SDK de transmisión. El SDK de un proveedor de filtros externos puede gestionar el ciclo de vida de la cámara para procesar las imágenes de la cámara, aplicar un efecto de filtro y emitirlas en un formato que se pueda pasar a una fuente de imágenes personalizada.



Consulte la documentación de su proveedor de filtros externos para conocer los métodos integrados para convertir un fotograma de cámara, con el efecto de filtro, aplicado a un formato que se pueda pasar a una [fuente de entrada de imágenes personalizada](#). El proceso varía según la versión del SDK de transmisión de IVS que se utilice:

- **Web:** el proveedor de filtros debe poder renderizar su salida en un elemento de lienzo. Se puede usar el método [CaptureStream](#) para devolver un `MediaStream` del contenido del lienzo. El `MediaStream` se puede convertir en una instancia de [LocalStageStream](#) y publicarse en un escenario.
- **Android:** el SDK del proveedor de filtros puede renderizar un marco en un dispositivo Android `Surface` proporcionado por el SDK de transmisión de IVS o convertir el marco en un mapa de bits. Si utiliza un mapa de bits, puede renderizarlo en el `Surface` subyacente proporcionado por la fuente de imagen personalizada, desbloqueándolo y escribiéndolo en un lienzo.

- iOS: el SDK de un proveedor de filtros externo debe proporcionar un marco de cámara con un efecto de filtro aplicado como `CMSampleBuffer`. Consulte la documentación del SDK de su proveedor de filtros externo para obtener información sobre cómo obtener un `CMSampleBuffer` como resultado final después de procesar una imagen de cámara.

Uso de BytePlus con el SDK de transmisión de IVS

Este documento explica cómo utilizar el SDK de BytePlus Effects con el SDK de transmisión de IVS.

Android

Instalación y configuración del SDK de BytePlus Effects

Consulte la [Guía de acceso de Android](#) para BytePlus para obtener detalles sobre cómo instalar, inicializar y configurar el SDK de BytePlus Effects.

Configuración de la fuente de imagen personalizada

Tras inicializar el SDK, alimente los fotogramas de cámara procesados con un efecto de filtro aplicado a una fuente de entrada de imágenes personalizada. Para ello, cree una instancia de un objeto `DeviceDiscovery` y cree una fuente de imagen personalizada. Observe que cuando utiliza una fuente de entrada de imagen personalizada para el control personalizado de la cámara, el SDK de transmisión ya no es responsable de administrar la cámara. En cambio, la aplicación es responsable de manejar correctamente el ciclo de vida de la cámara.

Java

```
var deviceDiscovery = DeviceDiscovery(applicationContext)
var customSource = deviceDiscovery.createImageInputSource( BroadcastConfiguration.Vec2(
    720F, 1280F
))
var surface: Surface = customSource.inputSurface
var filterStream = ImageLocalStageStream(customSource)
```

Convierta la salida en un mapa de bits y la transmisión en una fuente de entrada de imágenes personalizada

Para permitir que los fotogramas de cámara con un efecto de filtro aplicado desde el SDK de `BytePlusEffect` se reenvíen directamente al SDK de transmisión de IVS, convierta la salida de una textura del SDK de BytePlus Effects en un mapa de bits. Cuando se procesa una imagen, el SDK

invoca al método `onDrawFrame()`. El método `onDrawFrame()` es un método público de la interfaz [GLSurfaceView.Renderer](#) de Android. En la aplicación de ejemplo para Android proporcionada por BytePlus, este método se utiliza en todos los fotogramas de la cámara y genera una textura. Al mismo tiempo, puede complementar el método `onDrawFrame()` con la lógica para convertir esta textura en un mapa de bits y enviarla a una fuente de entrada de imágenes personalizada. Como se muestra en el siguiente ejemplo de código, utilice el método `transferTextureToBitmap` proporcionado por el SDK de BytePlus para realizar esta conversión. Este método lo proporciona la biblioteca [com.bytedance.labcv.core.util.ImageUtil](#) del SDK de BytePlus Effects, como se muestra en el siguiente ejemplo de código. Luego, puede renderizar el Surface subyacente para Android de `CustomImageSource` al escribir el mapa de bits resultante en el lienzo de Surface. Muchas invocaciones sucesivas de `onDrawFrame()` dan como resultado una secuencia de mapas de bits y, cuando se combinan, crean una secuencia de vídeo.

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);
}
```

Uso de DeepAR con el SDK de transmisión de IVS

Este documento explica cómo utilizar el SDK de DeepAR con el SDK de transmisión de IVS.

Android

Consulte [Android Integration Guide de DeepAR](#) para obtener detalles sobre cómo integrar el SDK de DeepAR con el SDK de retransmisión de Android IVS.

iOS

Consulte la [Guía de integración de iOS de DeepAR](#) para obtener más información sobre cómo integrar el SDK de DeepAR con el SDK de transmisión IVS para iOS.

Uso de Snap con el SDK de transmisión de IVS

Este documento explica cómo utilizar el SDK de Camera Kit de Snap con el SDK de transmisión de IVS.

Web

En esta sección se presupone que ya está familiarizado con la [publicación de vídeos y la suscripción a ellos mediante el SDK de transmisión web](#).

Para integrar el SDK de Camera Kit de Snap con el SDK de transmisión Web en tiempo real de IVS, debe:

1. Instale el SDK y el Webpack de Camera Kit. (Nuestro ejemplo usa Webpack como paquete, pero puede usar cualquier paquete de su elección).
2. Cree `index.html`.
3. Agregue elementos de configuración.
4. Muestre y configure los participantes.
5. Muestre las cámaras y los micrófonos conectados.
6. Cree una sesión de Camera Kit.
7. Busque y aplique una lente.
8. Renderice el resultado de una sesión de Camera Kit en un lienzo.
9. Proporcione a Camera Kit una fuente multimedia para renderizar y publicar `unLocalStageStream`.
10. Cree un archivo de configuración de Webpack.

A continuación, se describe cada uno de estos pasos.

Instalación del SDK y del Webpack de Camera Kit

```
npm i @snap/camera-kit webpack webpack-cli
```

Creación del archivo index.html

A continuación, cree la plantilla HTML e importe el SDK de transmisión web como una etiqueta script. En el código siguiente, asegúrese de reemplazar `<SDK version>` por la versión del SDK de transmisión que esté utilizando.

JavaScript

```
<!--
/*! Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved. SPDX-License-
Identifier: Apache-2.0 */
-->
<!DOCTYPE html>
<html lang="en">

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

  <title>Amazon IVS Real-Time Streaming Web Sample (HTML and JavaScript)</title>

  <!-- Fonts and Styling -->
  <link rel="stylesheet" href="https://fonts.googleapis.com/css?
family=Roboto:300,300italic,700,700italic" />
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/normalize/8.0.1/
normalize.css" />
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/milligram/1.4.1/
milligram.css" />
  <link rel="stylesheet" href="./index.css" />

  <!-- Stages in Broadcast SDK -->
  <script src="https://web-broadcast.live-video.net/<SDK version>/amazon-ivs-web-
broadcast.js"></script>
</head>

<body>
  <!-- Introduction -->
  <header>
    <h1>Amazon IVS Real-Time Streaming Web Sample (HTML and JavaScript)</h1>

    <p>This sample is used to demonstrate basic HTML / JS usage. <b><a href="https://
docs.aws.amazon.com/ivs/latest/LowLatencyUserGuide/multiple-hosts.html">Use the AWS
```

```

CLI to create a Stage and a corresponding ParticipantToken.
Multiple participants can load this page and put in their own tokens. You can read more about stages in our public docs.
</header>
<hr />

<!-- Setup Controls -->

<!-- Local Participant -->

<hr style="margin-top: 5rem"/>

<!-- Remote Participants -->

<!-- Load all Desired Scripts -->

</body>

</html>

```

Agregue elementos de configuración

Cree el código HTML para seleccionar una cámara y un micrófono, y especificar un token de participante:

JavaScript

```

<!-- Setup Controls -->
<div class="row">
  <div class="column">
    <label for="video-devices">Select Camera</label>
    <select disabled id="video-devices">
      <option selected disabled>Choose Option</option>
    </select>
  </div>
  <div class="column">
    <label for="audio-devices">Select Microphone</label>
    <select disabled id="audio-devices">
      <option selected disabled>Choose Option</option>
    </select>
  </div>
  <div 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>

```

Añada un código HTML adicional debajo para mostrar las imágenes de las cámaras de los participantes locales y remotos:

JavaScript

```

<!-- Local Participant -->
<div class="row local-container">
  <canvas id="canvas"></canvas>

  <div class="column" id="local-media"></div>
  <div class="static-controls hidden" id="local-controls">
    <button class="button" id="mic-control">Mute Mic</button>
    <button class="button" id="camera-control">Mute Camera</button>
  </div>
</div>

<hr style="margin-top: 5rem"/>

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

```

Cargue lógica adicional, incluidos los métodos de ayuda para configurar la cámara y el archivo JavaScript incluido. (Más adelante en esta sección, creará estos archivos JavaScript y los agrupará en un solo archivo para poder importar Camera Kit como un módulo. El archivo JavaScript agrupado contendrá la lógica para configurar Camera Kit, aplicar un lente y publicar la imagen de la cámara con un lente aplicado a un escenario).

JavaScript

```
<!-- Load all Desired Scripts -->
<script src="./helpers.js"></script>
<script src="./media-devices.js"></script>
<!-- <script type="module" src="./stages-simple.js"></script> -->
<script src="./dist/bundle.js"></script>
```

Visualización y configuración de los participantes

A continuación, cree `helpers.js`, que contiene métodos auxiliares que utilizará para mostrar y configurar los participantes:

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;
}
```

Visualización de las cámaras y los micrófonos conectados

A continuación, cree `media-devices.js`, que contiene métodos auxiliares para mostrar las cámaras y los micrófonos conectados a su dispositivo:

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,
  },
});
}

```

Creación de una sesión de Camera Kit

Cree `stages.js`, que contiene la lógica para aplicar una lente a la transmisión de la cámara y publicar la transmisión en un escenario. En la primera parte de este archivo, importamos el SDK de transmisión y el SDK web de Camera Kit e inicializamos las variables que usaremos con cada SDK. Para crear una sesión de Camera Kit, activamos `createSession` después de [impulsar el SDK web de Camera Kit](#). Tenga en cuenta que un objeto de elemento de lienzo se pasa a una sesión; esto le indica a Camera Kit que lo renderice en ese lienzo.

Java

```

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

// All helpers are expose on 'media-devices.js' and 'dom.js'
// const { setupParticipant } = window;
// const { initializeDeviceSelect, getCamera, getMic } = window;
// require('./helpers.js');
// require('./media-devices.js');

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

import {
  bootstrapCameraKit,
  createMediaStreamSource,
  Transform2D,
} from '@snap/camera-kit';

let cameraButton = document.getElementById('camera-control');
let micButton = document.getElementById('mic-control');

```

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

let controls = document.getElementById('local-controls');
let videoDevicesList = document.getElementById('video-devices');
let audioDevicesList = document.getElementById('audio-devices');

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

const liveRenderTarget = document.getElementById('canvas');

const init = async () => {
  await initializeDeviceSelect();

  const cameraKit = await bootstrapCameraKit({
    apiToken: INSERT_API_TOKEN_HERE,
  });

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

Búsqueda y aplicación de una lente

Para buscar sus lentes, introduzca su ID de grupo de lentes, el cual puede encontrarlo en el [portal para desarrolladores de Camera Kit](#). En este ejemplo, lo simplificamos aplicando la primera lente de la matriz de lentes que se devuelve.

JavaScript

```
const { lenses } = await cameraKit.lensRepository.loadLensGroups([
  INSERT_LENS_GROUP_ID_HERE,
]);

session.applyLens(lenses[0]);
```

Renderización del resultado de una sesión de Camera Kit en un lienzo

Utilice el método [CaptureStream](#) para devolver una `MediaStream` del contenido del lienzo. El lienzo contendrá una transmisión de vídeo de la imagen de la cámara con una lente aplicada. Además, añada oyentes de eventos como botones para silenciar la cámara y el micrófono, así como oyentes de eventos para entrar y salir del escenario. En el oyente de eventos para unirse a un escenario, pasamos una sesión de Camera Kit y la `MediaStream` del lienzo para poder publicarla en un escenario.

JavaScript

```
const snapStream = liveRenderTarget.captureStream();

cameraButton.addEventListener('click', () => {
  const isMuted = !cameraStageStream.isMuted;
  cameraStageStream.setMuted(isMuted);
  cameraButton.innerText = isMuted ? 'Show Camera' : 'Hide Camera';
});

micButton.addEventListener('click', () => {
  const isMuted = !micStageStream.isMuted;
  micStageStream.setMuted(isMuted);
  micButton.innerText = isMuted ? 'Unmute Mic' : 'Mute Mic';
});

joinButton.addEventListener('click', () => {
  joinStage(session, snapStream);
});

leaveButton.addEventListener('click', () => {
  leaveStage();
});
};
```

Proporcione a Camera Kit una fuente multimedia para renderizar y publique un `LocalStageStream`

Para publicar una transmisión de vídeo con una lente aplicada, cree una función llamada `setCameraKitSource` para transferir la `MediaStream` capturada anteriormente desde el lienzo. La `MediaStream` desde el lienzo no sirve de nada por el momento porque aún no hemos incorporado nuestra cámara local. Podemos incorporar la señal de nuestra cámara local llamando al método auxiliar `getCamera` y asignándolo a `localCamera`. Luego, podemos pasar la señal

de nuestra cámara local (vía `localCamera`) y el objeto de sesión a `setCameraKitSource`. La función `setCameraKitSource` convierte la señal de nuestra cámara local en una [fuente de contenido multimedia para Camera Kit](#) con solo una llamada a `createMediaStreamSource`. La fuente multimedia para `CameraKit` se [transforma](#) para reflejar la cámara frontal. El efecto de la lente se aplica a la fuente multimedia y se renderiza en el lienzo de salida mediante una llamada `session.play()`.

Ahora que la lente está aplicada a la `MediaStream` capturada desde el lienzo, podemos proceder a publicarla en un escenario. Para ello, creamos un `LocalStageStream` con las pistas de vídeo de `MediaStream`. Luego, se puede pasar una instancia de `LocalStageStream` a `StageStrategy` para publicarla.

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;
  },
};
```

El código restante que aparece a continuación sirve para crear y gestionar nuestro escenario:

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();
```

Creación de un archivo de configuración de Webpack

Cree `webpack.config.js` y agregue el siguiente código. Esto agrupa la lógica anterior para que pueda usar la declaración de importación para usar Camera Kit.

JavaScript

```
const path = require('path');
module.exports = {
  entry: ['./stage.js'],
  output: {
    filename: 'bundle.js',
    path: path.resolve(__dirname, 'dist'),
  },
};
```

Por último, ejecute `npm run build` para agrupar su JavaScript como se define en el archivo de configuración de Webpack. A continuación, podrá servir a HTML y JavaScript desde un servidor web. Por ejemplo, puede utilizar el servidor HTTP de Python de la línea de comandos como se indica a continuación:

```
# Run this from the command line and the directory containing index.html
python3 -m http.server -d ./
```

Abra el navegador y vaya a `http://localhost:8000` si utiliza el servidor HTTP de Python. A continuación, debería ver en pantalla el efecto AR de la lente de Snap que especificó anteriormente aplicado a la transmisión de la cámara.

Android

Para integrar el SDK de Camera Kit de Snap con el SDK de transmisión para Android de IVS, debe instalar el SDK de Camera Kit, inicializar una sesión de Camera Kit, aplicar una lente y enviar el resultado de la sesión de Camera Kit a la fuente de entrada de la imagen personalizada.

Para instalar el SDK de Camera Kit, añada lo siguiente al archivo `build.gradle` de su módulo. Sustituya `$cameraKitVersion` por la [última versión del SDK del Camera Kit](#).

Java

```
implementation "com.snap.camerakit:camerakit:$cameraKitVersion"
```

Inicialice y obtenga un `cameraKitSession`. Camera Kit también proporciona un práctico contenedor para las API [CameraX](#) de Android, por lo que no tiene que escribir una lógica complicada para usar CameraX con Camera Kit. Puede usar el objeto `CameraXImageProcessorSource` como [fuente](#) para [ImageProcessor](#), que le permite iniciar fotogramas de transmisión con vistas previas de la cámara.

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();
}
```

Búsqueda y aplicación de lentes

Puede configurar las lentes y su orden en el carrusel del [portal para desarrolladores de Camera Kit](#):

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);
            });
    });
```



```
    }));  
});
```

Para transmitir, envíe los fotogramas procesados al Surface subyacente de una fuente de imagen personalizada. Use un objeto `DeviceDiscovery` y cree un `CustomImageSource` para devolver un `SurfaceSource`. A continuación, puede renderizar el resultado de una sesión de `CameraKit` en el Surface subyacente proporcionado por el `SurfaceSource`.

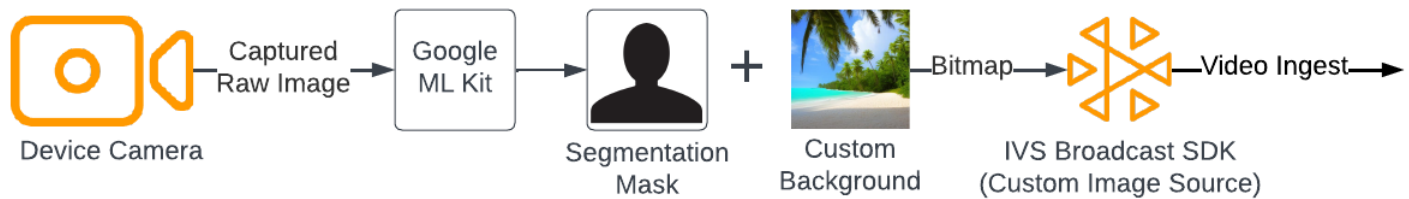
Java

```
val publishStreams = ArrayList<LocalStageStream>()  
  
val deviceDiscovery = DeviceDiscovery(applicationContext)  
val customSource =  
    deviceDiscovery.createImageInputSource(BroadcastConfiguration.Vec2(720f, 1280f))  
  
cameraKitSession.processor.connectOutput(outputFrom(customSource.inputSurface))  
val customStream = ImageLocalStageStream(customSource)  
  
// After rendering the output from a Camera Kit session to the Surface, you can  
// then return it as a LocalStageStream to be published by the Broadcast SDK  
val customStream: ImageLocalStageStream = ImageLocalStageStream(surfaceSource)  
publishStreams.add(customStream)  
  
@Override  
fun stageStreamsToPublishForParticipant(stage: Stage, participantInfo:  
    ParticipantInfo): List<LocalStageStream> = publishStreams
```

Uso del reemplazo de fondo con el SDK de transmisión de IVS

El reemplazo del fondo es un tipo de filtro de cámara que permite a los creadores de transmisiones en directo cambiar sus fondos. Como se muestra en el diagrama siguiente, reemplazar el fondo implica:

1. Obtener una imagen de cámara a partir de la transmisión de la cámara en directo.
2. Segmentarla en componentes de primer plano y segundo plano con el ML Kit de Google.
3. Combinar la máscara de segmentación resultante con una imagen de fondo personalizada.
4. Pasarla a una fuente de imagen personalizada para su transmisión.



Web

En esta sección se presupone que ya está familiarizado con la [publicación de vídeos y la suscripción a ellos mediante el SDK de transmisión web](#).

Para sustituir el fondo de una transmisión en directo por una imagen personalizada, utilice el [modelo de segmentación de selfies](#) con [MediaPipe Image Segmenter](#). Se trata de un modelo de machine learning que identifica qué píxeles del fotograma de vídeo están en primer plano o en segundo plano. Puede utilizar los resultados del modelo para sustituir el fondo de una transmisión en directo copiando los píxeles de primer plano de la transmisión de vídeo a una imagen personalizada que represente el nuevo fondo.

Para integrar el reemplazo del fondo de pantalla con el SDK de transmisión web en tiempo real de IVS, debe:

1. Instalar MediaPipe y Webpack. (Nuestro ejemplo usa Webpack como paquete, pero puede usar cualquier paquete de su elección).
2. Cree `index.html`.
3. Agregue elementos multimedia.
4. Añada una etiqueta de script.
5. Cree `app.js`.
6. Cargue una imagen de fondo personalizada.
7. Cree una instancia de `ImageSegmenter`.
8. Renderice la transmisión de vídeo en un lienzo.
9. Cree una lógica de reemplazo de fondo.
10. Cree el archivo de configuración de Webpack.
11. Agrupe su archivo JavaScript.

Instalación de MediaPipe y Webpack

Para empezar, instale `@mediapipe/tasks-vision` y los paquetes npm de webpack. El siguiente ejemplo usa Webpack como un empaquetador de JavaScript; puede usar un empaquetador diferente si lo prefiere.

JavaScript

```
npm i @mediapipe/tasks-vision webpack webpack-cli
```

Asegúrese de actualizar también su `package.json` para especificar webpack al desarrollar script:

JavaScript

```
"scripts": {
  "test": "echo \"Error: no test specified\" && exit 1",
  "build": "webpack"
},
```

Creación del archivo index.html

A continuación, cree la plantilla HTML e importe el SDK de transmisión web como una etiqueta script. En el código siguiente, asegúrese de reemplazar `<SDK version>` por la versión del SDK de transmisión que esté utilizando.

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>
```

Agregue elementos multimedia

A continuación, añada un elemento de vídeo y dos elementos de lienzo dentro de la etiqueta corporal. El elemento de vídeo contendrá las imágenes de la cámara en directo y se utilizará como entrada para el segmentador de imágenes de MediaPipe. El primer elemento de lienzo se utilizará para obtener una vista previa de la transmisión que se emitirá. El segundo elemento de lienzo se utilizará para renderizar la imagen personalizada que se utilizará como fondo. Como el segundo lienzo con la imagen personalizada solo se usa como fuente para copiar píxeles mediante programación al lienzo final, está oculto a la vista.

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>
```

Agregue una etiqueta de script

Agregue una etiqueta de script para cargar un archivo JavaScript agrupado que contendrá el código para reemplazar el fondo y publicarlo en un escenario:

```
<script src="./dist/bundle.js"></script>
```

Creación de app.js

A continuación, cree un archivo JavaScript para obtener los objetos de elemento para los elementos de lienzo y vídeo que se crearon en la página HTML. Importe los módulos `ImageSegmenter`

y `FilesetResolver`. El módulo `ImageSegmenter` se utilizará para realizar la tarea de segmentación.

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";
```

A continuación, cree una función llamada `init()` para recuperar el `MediaStream` de la cámara del usuario e invoque una función de devolución de llamada cada vez que termine de cargarse el fotograma de la cámara. Añada detectores de eventos para que los botones se unan y salgan de un escenario.

Tenga en cuenta que cuando nos unimos a un escenario, pasamos una variable llamada `segmentationStream`. Se trata de una transmisión de vídeo capturada desde un elemento de lienzo, que contiene una imagen de primer plano superpuesta a la imagen personalizada que representa el fondo. Más adelante, esta transmisión personalizada se utilizará para crear una instancia de `LocalStageStream`, que se podrá publicar en un escenario.

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();
});
};
```

Carga de una imagen de fondo personalizada

En la parte inferior de la función `init`, añada código para llamar a una función llamada `initBackgroundCanvas`, que carga una imagen personalizada de un archivo local y la renderiza en un lienzo. Definiremos esta función en el siguiente paso. Asigne la `MediaStream` recuperada de la cámara del usuario al objeto de vídeo. Más tarde, este objeto de vídeo se pasará al segmentador de imágenes. Además, configure una función denominada `renderVideoToCanvas` como la función de devolución de llamada para que se invoque cada vez que un fotograma de vídeo termine de cargarse. Definiremos esta función en un paso posterior.

JavaScript

```
initBackgroundCanvas();

video.srcObject = localCamera;
video.addEventListener("loadeddata", renderVideoToCanvas);
```

Vamos a implementar la función `initBackgroundCanvas`, que carga una imagen desde un archivo local. En este ejemplo, utilizaremos una imagen de una playa como fondo personalizado. El lienzo que contiene la imagen personalizada se ocultará de la pantalla, ya que lo combinará con los píxeles del primer plano del elemento del lienzo que contiene la imagen de la cámara.

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);
  };
};
```

```
};
};
```

Creación de una instancia de ImageSegmenter

A continuación, cree una instancia de `ImageSegmenter`, que segmentará la imagen y devolverá el resultado en forma de máscara. Al crear una instancia de `ImageSegmenter`, utilizará el [modelo de segmentación de selfies](#).

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,
  });
};
```

Renderice la transmisión de vídeo en un lienzo

A continuación, cree la función que renderice la transmisión de vídeo al otro elemento del lienzo. Necesitamos renderizar la transmisión de vídeo en un lienzo para poder extraer los píxeles del primer plano utilizando la API Canvas 2D. Mientras lo hacemos, también pasaremos un fotograma de vídeo a nuestra instancia `ImageSegmenter`, utilizando el método [segmentForVideo](#) para segmentar el primer plano del fondo del fotograma de vídeo. Cuando el método [segmentForVideo](#) regrese, invoque nuestra función de devolución de llamada personalizada, `replaceBackground`, para reemplazar el fondo.

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);
};

```

Creación de una lógica de reemplazo en segundo plano

Cree la función `replaceBackground`, que fusiona la imagen de fondo personalizada con el primer plano de la transmisión de la cámara para reemplazar el fondo. La función recupera primero los datos de píxeles subyacentes de la imagen de fondo personalizada y la transmisión de vídeo de los dos elementos del lienzo creados anteriormente. A continuación, recorre en iteración la máscara proporcionada por `ImageSegmenter`, que indica qué píxeles están en primer plano. A medida que recorre la máscara, copia de forma selectiva los píxeles que contienen la imagen de la cámara del usuario en los datos de píxeles de fondo correspondientes. Una vez hecho esto, convierte los datos de píxeles finales con el primer plano copiado en el fondo y los dibuja en un lienzo.

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);
}
```

Como referencia, aquí está el archivo `app.js` completo que contiene toda la lógica anterior:

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();
```

Creación de un archivo de configuración de Webpack

Agregue esta configuración a su archivo de configuración de Webpack para `empaquetarapp.js`, de modo que las llamadas de importación funcionen:

JavaScript

```
const path = require("path");
module.exports = {
  entry: ["/app.js"],
  output: {
    filename: "bundle.js",
    path: path.resolve(__dirname, "dist"),
  },
};
```

Agrupación de sus archivos JavaScript

```
npm run build
```

Inicie un servidor HTTP simple desde el directorio que contiene `index.html` y abra `localhost:8000` para ver el resultado:

```
python3 -m http.server -d ./
```

Android

Para reemplazar el fondo de tu transmisión en directo, puede usar la API de segmentación de selfies del [ML Kit de Google](#). La API de segmentación de selfies acepta una imagen de cámara como entrada y devuelve una máscara que proporciona una puntuación de confianza para cada píxel de la imagen, que indica si estaba en primer plano o en segundo plano. En función de la puntuación de confianza, puede recuperar el color de píxel correspondiente de la imagen de fondo o de la imagen de primer plano. Este proceso continúa hasta que se hayan examinado todas las puntuaciones de confianza de la máscara. El resultado es una nueva matriz de colores de píxeles que contiene los píxeles del primer plano combinados con los píxeles de la imagen de fondo.

Para integrar la sustitución en segundo plano con el SDK de retransmisión para transmisión en tiempo real de IVS:

1. Instale las bibliotecas CameraX y el kit ML de Google.
2. Inicialice las variables repetitivas.
3. Cree de una fuente de imágenes personalizada.
4. Administre los fotogramas de las cámaras.
5. Transfiera los marcos de las cámaras al ML Kit de Google.
6. Superponga el primer plano del marco de la cámara sobre su fondo personalizado.
7. Introduzca la nueva imagen en una fuente de imágenes personalizada.

Instalación de las bibliotecas CameraX y del kit ML de Google

Para extraer imágenes de la transmisión de la cámara en vivo, use la biblioteca CameraX de Android. Para instalar la biblioteca CameraX y el kit ML de Google, añada lo siguiente al archivo `build.gradle` del módulo. Sustituya `${camerax_version}` y `${google_ml_kit_version}` por la última versión de las bibliotecas [CameraX](#) y [ML Kit Google](#), respectivamente.

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}"
```

Importe las siguientes bibliotecas:

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
```

Inicialización de variables reutilizables

Inicialice una instancia de `ImageAnalysis` y una instancia de `ExecutorService`:

Java

```
private lateinit var binding: ActivityMainBinding
private lateinit var cameraExecutor: ExecutorService
private var analysisUseCase: ImageAnalysis? = null
```

[Inicialice una instancia en `STREAM_MODE`](#):

Java

```
private val options =
    SelfieSegmenterOptions.Builder()
        .setDetectorMode(SelfieSegmenterOptions.STREAM_MODE)
        .build()

private val segmenter = Segmentation.getClient(options)
```

Creación de una fuente de imágenes personalizada

En el método `onCreate` de su actividad, cree una instancia de un objeto `DeviceDiscovery` y cree una fuente de imágenes personalizada. El `Surface` proporcionado por la fuente de imagen personalizada recibirá la imagen final, con el primer plano superpuesto sobre una imagen de fondo

personalizada. A continuación, creará una instancia de `ImageLocalStageStream` utilizando la fuente de imagen personalizada. Luego, la instancia de `ImageLocalStageStream` (nombrada `filterStream` en este ejemplo) puede publicarse en un escenario. Consulte la [Guía del SDK de transmisión para Android de IVS](#) para obtener instrucciones sobre cómo configurar un escenario. Por último, cree también un hilo que se utilizará para gestionar la cámara.

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()
```

Administración de fotogramas de cámara

A continuación, cree una función para inicializar la cámara. Esta función utiliza la biblioteca `CameraX` para extraer imágenes de la transmisión de la cámara en directo. En primer lugar, se crea una instancia de `ProcessCameraProvider` llamada `cameraProviderFuture`. Este objeto representa un resultado futuro de la obtención de un proveedor de cámaras. A continuación, cargue una imagen del proyecto en forma de mapa de bits. En este ejemplo se utiliza la imagen de una playa como fondo, pero puede ser cualquier imagen que desee.

A continuación, añada un oyente a `cameraProviderFuture`. Este oyente recibe una notificación cuando la cámara está disponible o si se produce un error durante el proceso de obtención de un proveedor de cámaras.

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))
}
```

En el oyente, cree `ImageAnalysis.Builder` para acceder a cada fotograma individual de la transmisión de la cámara en directo. Establezca la estrategia de contrapresión para

STRATEGY_KEEP_ONLY_LATEST. Esto garantiza que solo se entregue un cuadro de cámara a la vez para su procesamiento. Convierta cada fotograma de cámara individual en un mapa de bits, de forma que pueda extraer sus píxeles y luego combinarlos con la imagen de fondo personalizada.

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())
```

Transferencia de los marcos de cámara al ML Kit de Google

A continuación, cree una `InputImage` y pásela a la instancia de `Segmenter` para su procesamiento. Se puede crear una `InputImage` a partir de una `ImageProxy` proporcionada por la instancia de `ImageAnalysis`. Una vez que se proporciona una `InputImage` a `Segmenter`, devuelve una máscara con puntuaciones de confianza que indican la probabilidad de que un píxel esté en primer plano o en segundo plano. Esta máscara también proporciona propiedades de ancho y alto, que utilizará para crear una nueva matriz que contenga los píxeles de fondo de la imagen de fondo personalizada cargada anteriormente.

Java

```
if (mediaImage != null) {
    val inputImage =
        InputImage.fromMediaImage

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)
```

Superposición del primer plano del marco de la cámara sobre el fondo personalizado

Con la máscara que contiene las puntuaciones de confianza, el marco de la cámara como mapa de bits y los píxeles de color de la imagen de fondo personalizada, tiene todo lo que necesita para superponer el primer plano al fondo personalizado. A continuación, se invoca la función `overlayForeground` con los siguientes parámetros:

Java

```
resultBitmap = overlayForeground(mask, maskWidth, maskHeight, inputBitmap,
    backgroundPixels)
```

Esta función recorre la máscara y comprueba los valores de confianza para determinar si se debe obtener el color de píxel correspondiente de la imagen de fondo o del marco de la cámara. Si el valor de confianza indica que lo más probable es que un píxel de la máscara esté en segundo plano, obtendrá el color de píxel correspondiente de la imagen de fondo; de lo contrario, obtendrá el color de píxel correspondiente del marco de la cámara para crear el primer plano. Una vez que la función termine de recorrer la máscara, se crea un nuevo mapa de bits con la nueva matriz de píxeles de color y se devuelve. Este nuevo mapa de bits contiene el primer plano superpuesto sobre el fondo personalizado.

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
)
}

```

Introducción de la nueva imagen en una fuente de imagen personalizada

A continuación, puede escribir el nuevo mapa de bits en el Surface proporcionado por una fuente de imagen personalizada. Esto se transmitirá a su escenario.

Java

```

resultBitmap = overlayForeground(mask, inputBitmap, mutableBitmap, bgBitmap)
canvas = surface.lockCanvas(null);
canvas.drawBitmap(resultBitmap, 0f, 0f, null)

```

Esta es la función completa para obtener los fotogramas de la cámara, pasarlos a Segmenter y superponerlos sobre el fondo:

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 de transmisión de IVS: modos de audio móvil | Transmisión en tiempo real

La calidad del audio es una parte importante de la experiencia multimedia de cualquier equipo real, y no existe una configuración de audio única que funcione mejor para todos los casos de uso. Para garantizar que sus usuarios disfruten de la mejor experiencia al escuchar una transmisión en tiempo real de IVS, nuestros SDK para dispositivos móviles ofrecen varias configuraciones de audio predefinidas, así como personalizaciones más potentes, según sea necesario.

Introducción

Los SDK de transmisión móvil de IVS ofrecen una clase `StageAudioManager`. Esta clase está diseñada para ser el único punto de contacto para controlar los modos de audio subyacentes en ambas plataformas. En Android, esto controla el [AudioManager](#), incluidos el modo de audio, la fuente de audio, el tipo de contenido, el uso y los dispositivos de comunicación. En iOS, controla la aplicación [AVAudioSession](#), así como si [VoiceProcessing](#) está habilitado.

Importante: no interactúe con `AVAudioSession` ni `AudioManager` directamente mientras el SDK de transmisión en tiempo real de IVS esté activo. Si lo hace, podría perderse el audio o grabarse o reproducirse en el dispositivo incorrecto.

Antes de crear el primer objeto `DeviceDiscovery` o `Stage`, debe estar configurada la clase `StageAudioManager`.

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
```

Si no hay nada establecido en `StageAudioManager` antes de la inicialización de una instancia `DeviceDiscovery` o `Stage`, el ajuste preestablecido `VideoChat` se aplica automáticamente.

Modo preestablecido de audio

El SDK de transmisión en tiempo real proporciona tres ajustes preestablecidos, cada uno adaptado a los casos de uso más comunes, tal y como se describe a continuación. Para cada ajuste preestablecido, cubrimos cinco categorías clave que diferencian los ajustes preestablecidos entre sí.

La categoría `Control de volumen` hace referencia al tipo de volumen (volumen de contenido multimedia o volumen de llamadas) que se utiliza o se cambia mediante los controles de volumen físicos del dispositivo. Tenga en cuenta que esto afecta al volumen al cambiar de modo de audio. Por ejemplo, supongamos que el volumen del dispositivo está ajustado al valor máximo mientras se utiliza el ajuste preestablecido de chat de video. Al cambiar al ajuste preestablecido `Suscripción única`, el nivel de volumen es diferente al del sistema operativo, lo que podría provocar un cambio de volumen significativo en el dispositivo.

Chat de vídeo

Este es el ajuste preestablecido predeterminado, diseñado para cuando el dispositivo local va a mantener una conversación en tiempo real con otros participantes.

Problema conocido en iOS: al usar este ajuste preestablecido y no conectar un micrófono, el audio se reproduce a través del auricular en lugar del altavoz del dispositivo. Use este ajuste preestablecido solo en combinación con un micrófono.

Categoría	Android	iOS
Cancelación de eco	Habilitado	Habilitado
Control de volumen	Volumen de llamadas	Volumen de llamadas
Selección de micrófono	Limitado según el sistema operativo. Es posible que los micrófonos USB no estén disponibles.	Limitado según el sistema operativo. Es posible que los micrófonos USB y Bluetooth no estén disponibles. Deberían funcionar los auriculares Bluetooth que admiten tanto la entrada como la salida al mismo tiempo; por ejemplo, los AirPods.
Salida de audio	Cualquier dispositivo de salida debería funcionar.	Limitado según el sistema operativo. Es posible que los auriculares con cable no estén disponibles.
Calidad de audio	Media/baja. Sonará como una llamada telefónica, no como una reproducción multimedia.	Media/baja. Sonará como una llamada telefónica, no como una reproducción multimedia.

Suscripción única

Este ajuste preestablecido está diseñado para suscribir a otros participantes de la publicación, pero no para publicar usted mismo. Se centra en la calidad del audio y es compatible con todos los dispositivos de salida disponibles.

Categoría	Android	iOS
Cancelación de eco	Deshabilitad	Deshabilitad
Control de volumen	Volumen multimedia	Volumen multimedia
Selección de micrófono	N/A, este ajuste preestablecido no está diseñado para su publicación.	N/A, este ajuste preestablecido no está diseñado para su publicación.
Salida de audio	Cualquier dispositivo de salida debería funcionar.	Cualquier dispositivo de salida debería funcionar.
Calidad de audio	Alta. Cualquier tipo multimedia debería mostrarse con claridad, incluida la música.	Alta. Cualquier tipo multimedia debería mostrarse con claridad, incluida la música.

Studio

Este ajuste preestablecido está diseñado para una suscripción de alta calidad y, al mismo tiempo, permite publicar. Requiere el hardware de grabación y reproducción para poder cancelar el eco. Un caso de uso en este caso sería usar un micrófono USB y unos auriculares con cable. El SDK mantendrá la más alta calidad de audio y, al mismo tiempo, se basará en la separación física de esos dispositivos para evitar que produzcan eco.

Categoría	Android	iOS
Cancelación de eco	Deshabilitad	Deshabilitad
Control de volumen	Volumen multimedia en la mayoría de los casos. Volumen de llamadas cuando hay un micrófono Bluetooth conectado.	Volumen multimedia
Selección de micrófono	Cualquier micrófono debería funcionar.	Cualquier micrófono debería funcionar.
Salida de audio	Cualquier dispositivo de salida debería funcionar.	Cualquier dispositivo de salida debería funcionar.

Categoría	Android	iOS
Calidad de audio	<p>Alta. Ambos lados deberían poder enviar música y escucharla con claridad en el otro lado.</p> <p>Cuando se conecta un auricular Bluetooth, la calidad del audio disminuirá debido a que el modo Bluetooth SCO está activado.</p>	<p>Alta. Ambos lados deberían poder enviar música y escucharla con claridad en el otro lado.</p> <p>Cuando se conecta un auricular Bluetooth, la calidad del audio puede disminuir debido a la activación del modo Bluetooth SCO, en función del auricular.</p>

Casos de uso avanzados

Más allá de los ajustes preestablecidos, los SDK de transmisión en tiempo real de iOS y Android permiten configurar los modos de audio de la plataforma subyacente:

- En Android, configure [AudioSource](#), [Usage](#) y [ContentType](#).
- En iOS, use [AVAudioSession.Category](#), [AVAudioSession.CategoryOptions](#), [AVAudioSession.Mode](#) y la posibilidad de activar o no el [procesamiento de voz](#) durante la publicación.

Nota: Al utilizar estos métodos del SDK de audio, es posible configurar incorrectamente la sesión de audio subyacente. Por ejemplo, si se utiliza la opción `.allowBluetooth` en iOS en combinación con la categoría `.playback`, se crea una configuración de audio no válida y el SDK no puede grabar ni reproducir audio. Estos métodos están diseñados para usarse solo cuando una aplicación tiene requisitos específicos de sesión de audio que se han validado.

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
```

Cancelación de eco de iOS

La cancelación de eco de iOS también se puede controlar de forma independiente a través de `IVSStageAudioManager` y mediante el método `echoCancellationEnabled`. Este método controla si el [procesamiento de voz](#) está habilitado en los nodos de entrada y salida de la propiedad subyacente `AVAudioEngine` utilizada por el SDK. Es importante comprender el efecto de cambiar esta propiedad de forma manual:

- La propiedad `AVAudioEngine` solo se respeta si el micrófono del SDK está activo; esto es necesario debido al requisito de iOS de que el procesamiento de voz esté habilitado simultáneamente en los nodos de entrada y salida. Normalmente, esto se hace con el micrófono devuelto por `IVSDeviceDiscovery` para crear un objeto `IVSLocalStageStream` y publicarlo. Como alternativa, el micrófono se puede habilitar, sin que se utilice para publicar, mediante la asociación de un objeto `IVSAudioDeviceStatsCallback` al propio micrófono. Este enfoque alternativo resulta útil si se necesita cancelar el eco al utilizar un micrófono basado en un origen de audio personalizado en lugar del micrófono del SDK del IVS.
- Para habilitar la propiedad `AVAudioEngine`, se requiere el modo `.videoChat` o `.voiceChat`. Al solicitar otro modo, la estructura de audio subyacente de iOS se opone al SDK, lo que provoca la pérdida de audio.
- Al habilitar `AVAudioEngine`, la opción `.allowBluetooth` se habilita automáticamente.

Los comportamientos pueden variar según el dispositivo y la versión de iOS.

Orígenes de audio personalizados para iOS

Los orígenes de audio personalizados se pueden usar con el SDK mediante `IVSDeviceDiscovery.createAudioSource`. Al conectarse a una fase, el SDK de retransmisión en tiempo real de IVS sigue administrando una instancia de `AVAudioEngine` interna para la reproducción de audio, aunque no se utilice el micrófono del SDK. Como resultado, los valores proporcionados a `IVSStageAudioManager` deben ser compatibles con el audio que proporciona el origen de audio personalizado.

Si el origen de audio personalizado que se utiliza para publicar graba desde el micrófono pero lo administra la aplicación host, el SDK de cancelación de eco anterior no funcionará a menos que el micrófono administrado por el SDK esté activado. Para evitar ese requisito, consulte [Cancelación de eco de iOS](#).

Publicación con Bluetooth en Android

El SDK vuelve automáticamente al `VIDEO_CHAT` preestablecido en Android cuando se cumplen las siguientes condiciones:

- La configuración asignada no usa el valor de uso `VOICE_COMMUNICATION`.
- Hay un micrófono Bluetooth conectado al dispositivo.
- El participante local está publicando en un escenario.

Esta es una limitación del sistema operativo Android en lo que respecta al uso de auriculares Bluetooth para grabar audio.

Integración con otros servicios de SDK

Como tanto iOS como Android admiten solo un modo de audio activo por aplicación, es habitual que surjan conflictos si la aplicación utiliza varios SDK que requieren el control del modo de audio. Cuando se encuentre con estos conflictos, puede probar algunas estrategias de resolución habituales, que se explican a continuación.

Coincidencia de los valores del modo de audio

Con las opciones de configuración de audio avanzadas del SDK de IVS o las funciones de otro SDK, haga que los dos SDK se alineen con los valores subyacentes.

Ágora

iOS

En iOS, decirle al SDK de Agora que mantenga la `AVAudioSession` activa evitará que se desactive mientras el SDK de transmisión en tiempo real de IVS lo esté utilizando.

```
myRtcEngine.SetParameters("{\"che.audio.keep.audiosession\":true}");
```

Android

Evita llamar `setEnabledSpeakerphone` en `RtcEngine`, y llame a `enableLocalAudio(false)` mientras publica con el SDK de transmisión en tiempo real de IVS. Puede volver a llamar a `enableLocalAudio(true)` cuando el SDK de IVS no esté publicándose.

Uso de Amazon EventBridge con la transmisión en tiempo real de IVS

Puede utilizar Amazon EventBridge para monitorear sus transmisiones de Amazon Interactive Video Service (IVS).

Amazon IVS envía eventos de cambio sobre el estado de sus transmisiones a Amazon EventBridge. Todos los eventos que se entregan son válidos. Sin embargo, los eventos se envían en la medida de lo posible, lo que significa que no hay garantía de que:

- Los eventos se entregan: puede producirse un evento designado (por ejemplo, se publica un participante), pero es posible que Amazon IVS no envíe un evento correspondiente a EventBridge. Amazon IVS intenta entregar eventos durante varias horas antes de darse por vencido.
- Los eventos que se entregan llegarán en un periodo de tiempo especificado: es posible que reciba eventos de hasta unas pocas horas de antigüedad.
- Los eventos se entregan en orden: los eventos pueden estar desordenados, especialmente si se envían con poco tiempo de diferencia. Por ejemplo, puede ver un participante no publicado antes que un participante publicado.

Si bien es raro que los eventos falten, se retrasen o estén fuera de secuencia, debe gestionar estas posibilidades si escribe programas críticos para el negocio que dependen del orden o la existencia de los eventos de notificación.

Puede crear reglas de EventBridge para cualquiera de los siguientes eventos.

Tipo de evento	Evento	Enviado cuando...
Cambio de estado de composición del IVS	Error en el destino	Falló un intento de enviar la salida a un destino. Por ejemplo, se produjo un error en la transmisión a un canal porque no había ninguna clave de transmisión o se estaba produciendo otra transmisión.
Cambio de estado de composición del IVS	Inicio de destino	La salida a un destino se inició correctamente.

Tipo de evento	Evento	Enviado cuando...
Cambio de estado de composición del IVS	Fin de destino	La salida a un destino ha finalizado.
Cambio de estado de composición del IVS	Reconexión de destino	Se interrumpió la salida a un destino y se está intentando volver a conectar.
Cambio de estado de composición del IVS	Inicio de la sesión	Se creó una sesión de Composición. Este evento se desencadena cuando una canalización de un proceso de Composición se inicializa correctamente. En este momento, el proceso de composición se ha suscrito correctamente a un Escenario y recibe contenido multimedia y puede componer vídeo.
Cambio de estado de composición del IVS	Fin de la sesión	Se ha completado una sesión de composición.
Cambio de estado de composición del IVS	Error de sesión	No se pudo inicializar una canalización de Composición debido a que los recursos de Escenario no estaban disponibles o a algún otro error interno.
Cambio en el estado de la grabación del participante de IVS	Inicio de la grabación	Un publicador se ha conectado a la fase y se está grabando en S3.
Cambio en el estado de la grabación del participante de IVS	Fin de la grabación	Un publicador se ha desconectado de la fase y todos los archivos restantes se han escrito en S3.

Tipo de evento	Evento	Enviado cuando...
Cambio en el estado de la grabación del participante de IVS	Error al iniciar la grabación	Un publicador se conecta a la fase, pero la grabación no se inicia debido a errores (por ejemplo, el bucket de S3 no existe o no está en la región correcta). La transmisión en directo de este publicador no se graba
Cambio en el estado de la grabación del participante de IVS	Error al finalizar la grabación	La grabación termina con un error debido a fallas detectadas durante el proceso (por ejemplo, si se produce un error al intentar escribir la lista de reproducción de contenido multimedia). Aún pueden escribirse algunos objetos en la ubicación de almacenamiento configurada.
Actualización de una fase de IVS	Participante publicado	Un participante comienza a publicar en una fase.
Actualización de una fase de IVS	Participante no publicado	Un participante ha dejado de publicar en una fase.
Actualización de una fase de IVS	Error de publicación del participante	El intento de un participante de publicar en una fase dio error.

Creación de reglas de Amazon EventBridge para Amazon IVS

Puede crear una regla que se active en función de un evento que emita Amazon IVS. Siga los pasos que se indican en [Crear una regla en Amazon EventBridge](#) en la Guía del usuario de Amazon EventBridge. Cuando seleccione un servicio, elija Interactive Video Service (IVS).

Ejemplos: cambio del estado de la composición

Error de destino: este evento se envía cuando se produce un error al intentar enviarlo a un destino. Por ejemplo, la transmisión a un canal falló porque no había ninguna clave de transmisión o se estaba produciendo otra transmisión.

```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-012345678901",
  "detail-type": "IVS Composition State Change",
  "source": "aws.ivs",
  "account": "aws_account_id",
  "time": "2017-06-12T10:23:43Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:ivs:us-east-1:aws_account_id:composition/123456789012"
  ],
  "detail": {
    "event_name": "Destination Failure",
    "stage_arn": "<stage-arn>",
    "id": "<Destination-id>",
    "reason": "eg. stream key invalid"
  }
}
```

Inicio de destino: este evento se envía cuando la salida a un destino se inicia correctamente.

```
{
  "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>",
  }
}
```

```
}
}
```

Final de destino: este evento se envía cuando finaliza la salida a un destino.

```
{
  "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>",
  }
}
```

Reconexión de destino: este evento se envía cuando se interrumpe la salida a un destino y se intenta volver a conectar.

```
{
  "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>",
  }
}
```

```
}
```

Inicio de sesión: este evento se envía cuando se crea una sesión de composición. Este evento se desencadena cuando una canalización de un proceso de composición se inicializa correctamente. En este momento, el proceso de composición se ha suscrito correctamente a un escenario y recibe contenido multimedia y puede componer vídeo.

```
{
  "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>"
  }
}
```

Fin de sesión: este evento se envía cuando se completa una sesión de composición y se eliminan todos los recursos.

```
{
  "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>"
  }
}
```

```
}

```

Error de sesión: este evento se envía cuando una canalización de composición no se pudo inicializar debido a que los recursos del escenario no estaban disponibles, no había participantes en el escenario o a algún otro error interno.

```
{
  "version": "0",
  "id": "01234567-0123-0123-0123-012345678901",
  "detail-type": "IVS Composition State Change",
  "source": "aws.ivs",
  "account": "aws_account_id",
  "time": "2017-06-12T10:23:43Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:ivs:us-east-1:aws_account_id:composition/123456789012"
  ],
  "detail": {
    "event_name": "Session Failure",
    "stage_arn": "<stage-arn>",
    "reason": "eg. no participants in the stage"
  }
}
```

Ejemplos: cambio en el estado de grabación de un participante individual

Inicio de la grabación: este evento se envía cuando un publicador se ha conectado a la fase y se está grabando en 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"
  }
}

```

Fin de la grabación: este evento se envía cuando un publicador se ha desconectado de la fase y todos los archivos restantes se han escrito en 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
  }
}

```

Error al iniciar la grabación: este evento se envía cuando un publicador se conecta a la fase, pero la grabación no se inicia debido a errores (por ejemplo, el bucket de S3 no existe o no está en la región correcta). La transmisión en directo del publicador no se graba.

```

{
  "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 al finalizar la grabación: este evento se envía cuando la grabación termina con un error debido a fallas detectadas durante el proceso (por ejemplo, si se produce un error al intentar escribir una lista de reproducción maestra). Aún pueden escribirse algunos objetos en la ubicación de almacenamiento configurada.

```

{
  "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
  }
}

```

Ejemplos: actualización de una fase

Los eventos de actualización de fases incluyen un nombre de evento (que clasifica el evento) y metadatos sobre el evento. Los metadatos incluyen el ID del participante que activó el evento, los ID de fase y sesión asociados y el ID de usuario.

Participante publicado: este evento se envía cuando un participante comienza a publicar en una fase.

```
{
  "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",
    "user_id": "Your User Id",
    "participant_id": "xYz1c2d3e4f"
  }
}
```

Participante no publicado: este evento se envía cuando un participante ha dejado de publicar en una fase.

```
{
  "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",
    "user_id": "Your User Id",
    "participant_id": "xYz1c2d3e4f"
  }
}
```

Error de publicación del participante: este evento se envía cuando el intento de un participante de publicar en una fase falla.

```
{
  "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"
  }
}
```

Composición del servidor de IVS | Transmisión en tiempo real

La composición del servidor utiliza un servidor IVS para mezclar el audio y el vídeo de todos los participantes del escenario y, a continuación, envía este vídeo mixto a un canal IVS (por ejemplo, para llegar a un público más amplio) o a un bucket de S3. La composición del servidor se invoca a través de los puntos de conexión del plano de control del IVS en la región de origen del escenario.

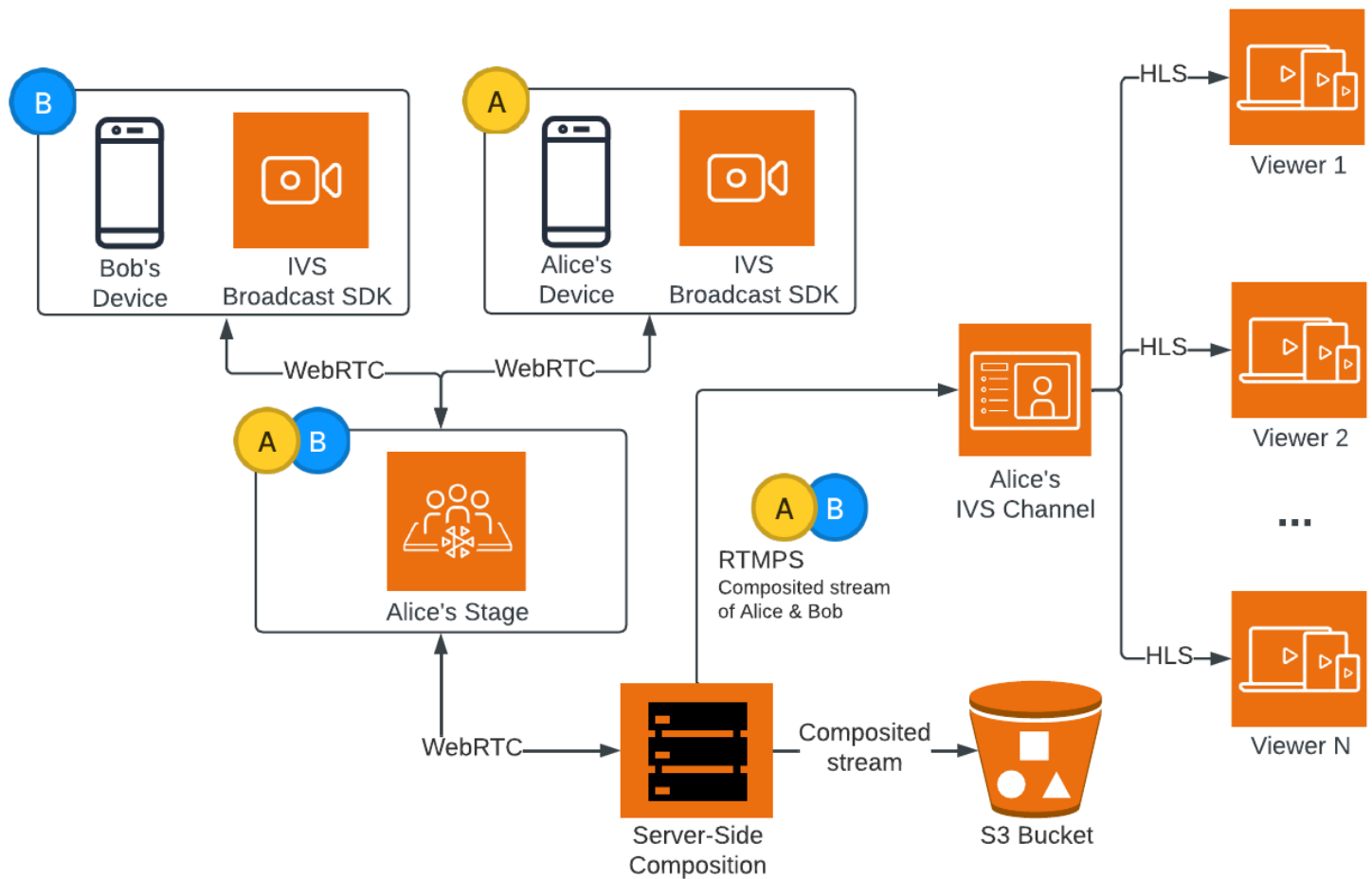
Transmitir o grabar un escenario utilizando la composición del servidor ofrece numerosas ventajas, lo que lo convierte en una opción atractiva para los usuarios que buscan flujos de trabajo de vídeo eficientes y fiables basados en la nube.

Temas

- [Información general sobre la composición del servidor de IVS](#)
- [Introducción a la composición del servidor de IVS](#)
- [Habilitación de la pantalla compartida en la composición del servidor de IVS](#)

Información general sobre la composición del servidor de IVS

Este diagrama ilustra cómo funciona la composición del servidor:



Ventajas

En comparación con la composición del cliente, la composición del servidor tiene las siguientes ventajas:

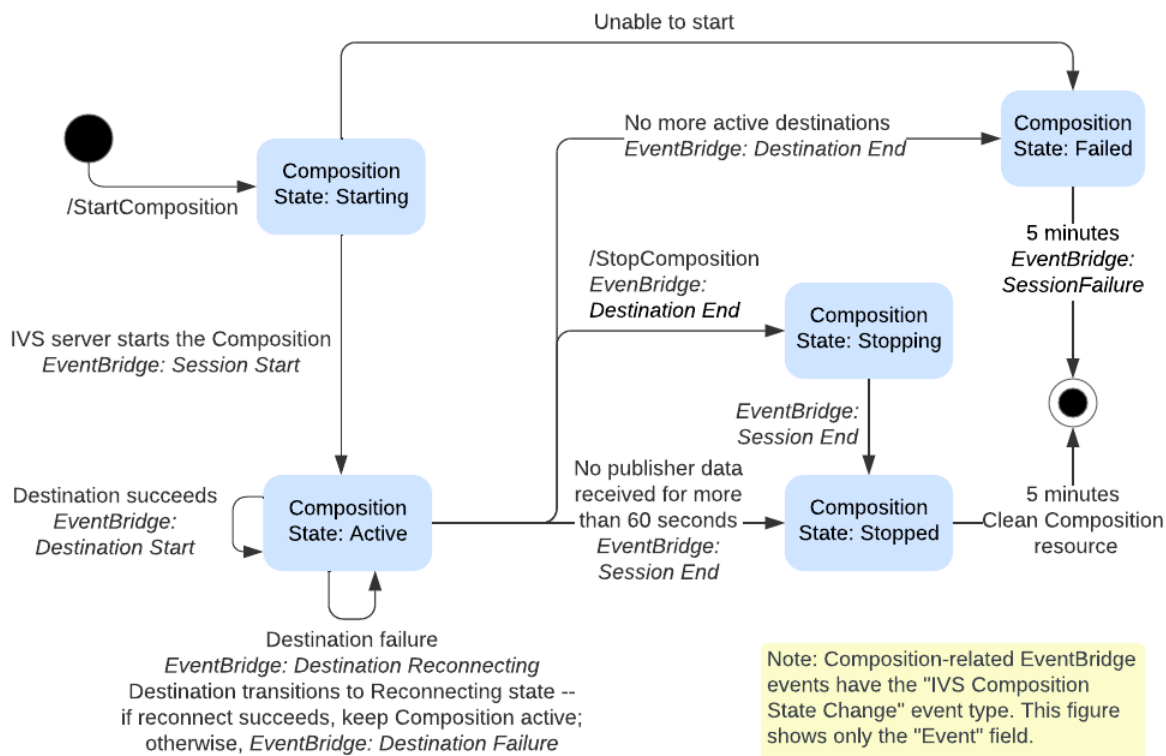
- **Reducción de la carga de clientes:** con la composición del servidor, la carga de procesar y combinar las fuentes de audio y vídeo pasa de los dispositivos cliente individuales al propio servidor. La composición del servidor elimina la necesidad de que los dispositivos cliente utilicen sus recursos de CPU y red para componer la vista y transmitirla al IVS. Esto significa que los espectadores pueden ver la transmisión sin que sus dispositivos tengan que realizar tareas que consumen muchos recursos, lo que puede mejorar la duración de la batería y disfrutar de una experiencia de visualización más fluida.
- **Calidad uniforme:** la composición del servidor permite un control preciso de la calidad, la resolución y la velocidad de bits de la transmisión final. Esto garantiza una experiencia de visualización uniforme para todos los espectadores, independientemente de las capacidades de sus dispositivos individuales.

- **Resiliencia:** al centralizar el proceso de composición en el servidor, la transmisión se vuelve más sólida. Incluso si el dispositivo de un publicador experimenta limitaciones o fluctuaciones técnicas, el servidor puede adaptarse y ofrecer una transmisión más fluida a toda la audiencia.
- **Eficiencia del ancho de banda:** dado que el servidor se encarga de la composición, los publicadores de la presentación no tienen que gastar más ancho de banda para transmitir el vídeo a IVS.

Como alternativa, para retransmitir una escena a un canal de IVS, puede realizar la composición desde el lado del cliente; consulte [Habilitación de varios hosts en una transmisión de Amazon IVS](#) en la Guía del usuario de transmisión de baja latencia de IVS.

Vida útil de la Composición

Utilice el siguiente diagrama para comprender las transiciones de estado de una composición:



A grandes rasgos, el ciclo de vida de una Composición es el siguiente:

1. Se crea un recurso de Composición cuando el usuario llama al punto de conexión `StartComposition`

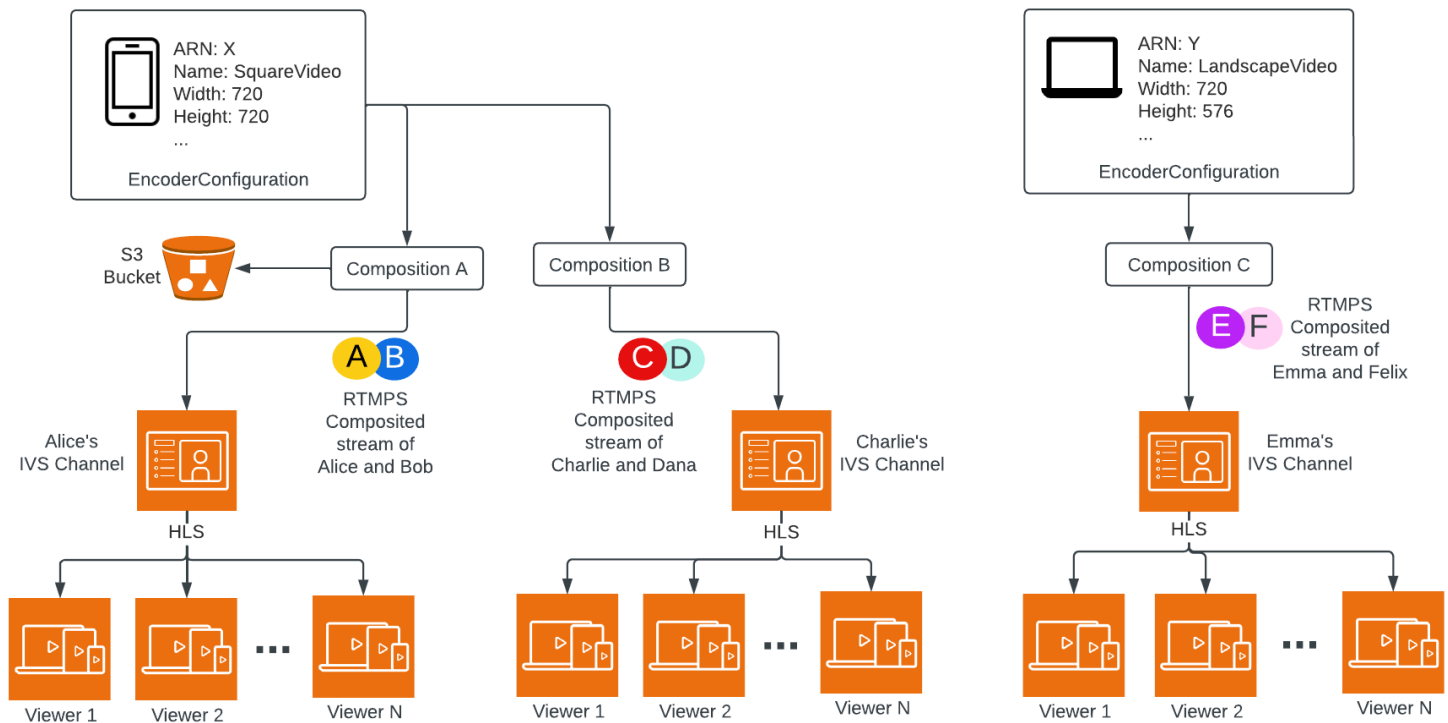
2. Una vez que IVS inicie correctamente la composición, se enviará un evento EventBridge de “Cambio de estado de composición de IVS (inicio de sesión)”. Consulte [Uso de EventBridge con la transmisión en tiempo real de IVS](#) para obtener más información sobre los eventos.
3. Una vez que una Composición está en estado activo, puede ocurrir lo siguiente:
 - Detención de la Composición por parte del usuario: si se llama al punto de conexión de StopComposition, IVS inicia un cierre correcto de la Composición y envía eventos de “Final de destino” seguidos de un evento de “Fin de sesión”.
 - Cierre automático de la Composición: si ningún participante publica activamente en la etapa IVS, la Composición finaliza automáticamente después de 60 segundos y se envían los eventos de EventBridge.
 - Error en el destino: si un destino falla inesperadamente (por ejemplo, si se elimina el canal IVS), el destino pasa al estado RECONNECTING y se envía un evento de “Reconexión del destino”. Si la recuperación no es posible, el IVS cambia el destino al estado FAILED y se envía un evento de “Error en el destino”. El IVS mantiene viva la composición si al menos uno de sus destinos está activo.
4. Una vez que la composición está en el estado STOPPED o FAILED, se limpia automáticamente después de cinco minutos. (Entonces, ListCompositions o GetComposition ya no la recuperan).

API de IVS

La composición del servidor utiliza estos elementos clave de la API:

- Un objeto EncoderConfiguration permite personalizar el formato del vídeo que se va a generar (altura, anchura, velocidad de bits y otros parámetros de transmisión). Puede reutilizar un EncoderConfiguration cada vez que llame al punto de conexión StartComposition.
- Los puntos de conexión de Composición rastrean la composición del vídeo y lo envían a un canal IVS.
- StorageConfiguration rastrea el bucket de S3 en el que se graban las composiciones.

Para utilizar la composición del servidor, debe crear una EncoderConfiguration y adjuntarla al llamar al punto de conexión StartComposition. En este ejemplo, la configuración de SquareVideo EncoderConfiguration se utiliza en dos composiciones:



Para obtener información completa, consulte la [Referencia de la API de transmisión en tiempo real de IVS](#).

Diseños

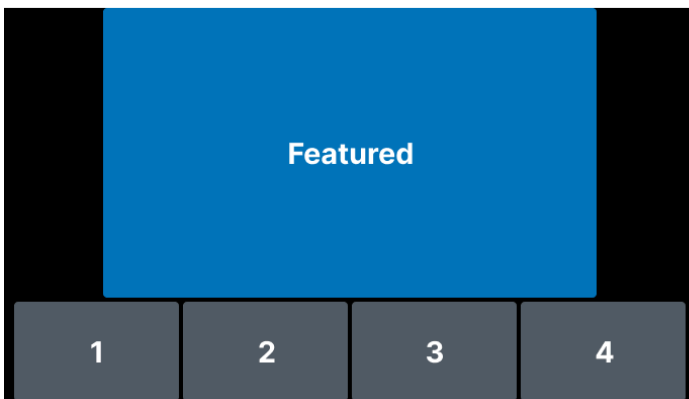
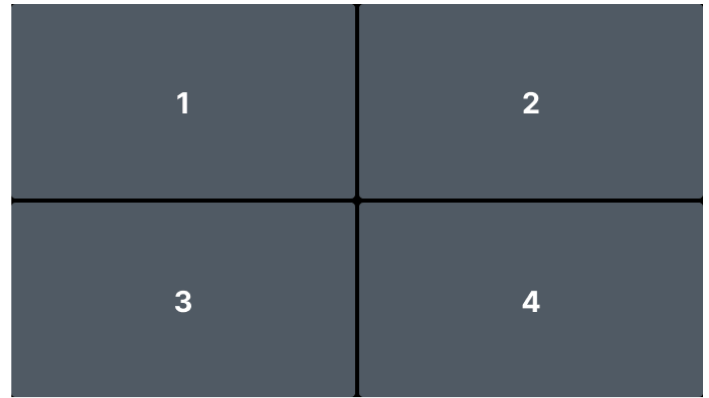
El punto de conexión StartComposition ofrece dos opciones de diseño: cuadrícula e imagen en imagen.

Diseño de cuadrícula

El diseño de cuadrícula organiza a los participantes de la fase en una cuadrícula de espacios del mismo tamaño. Proporciona varios beneficios personalizables:

- `videoAspectRatio` establece el modo de visualización de los participantes para controlar la relación de aspecto de los mosaicos de video.
- `videoFillMode` define cómo encaja el contenido de video en el mosaico del participante.
- `gridGap` especifica el espacio entre los mosaicos de los participantes en píxeles.
- `omitStoppedVideo` permite excluir de la composición las transmisiones de video detenidas.
- `featuredParticipantAttribute` identifica el espacio destacado. Cuando se establece, el participante destacado aparece en un espacio más grande en la pantalla principal y debajo aparecen los demás participantes.

Para obtener más información sobre el diseño de cuadrícula (incluidos los valores válidos y los valores predeterminados de todos los campos), consulte el tipo de datos [GridConfiguration](#).



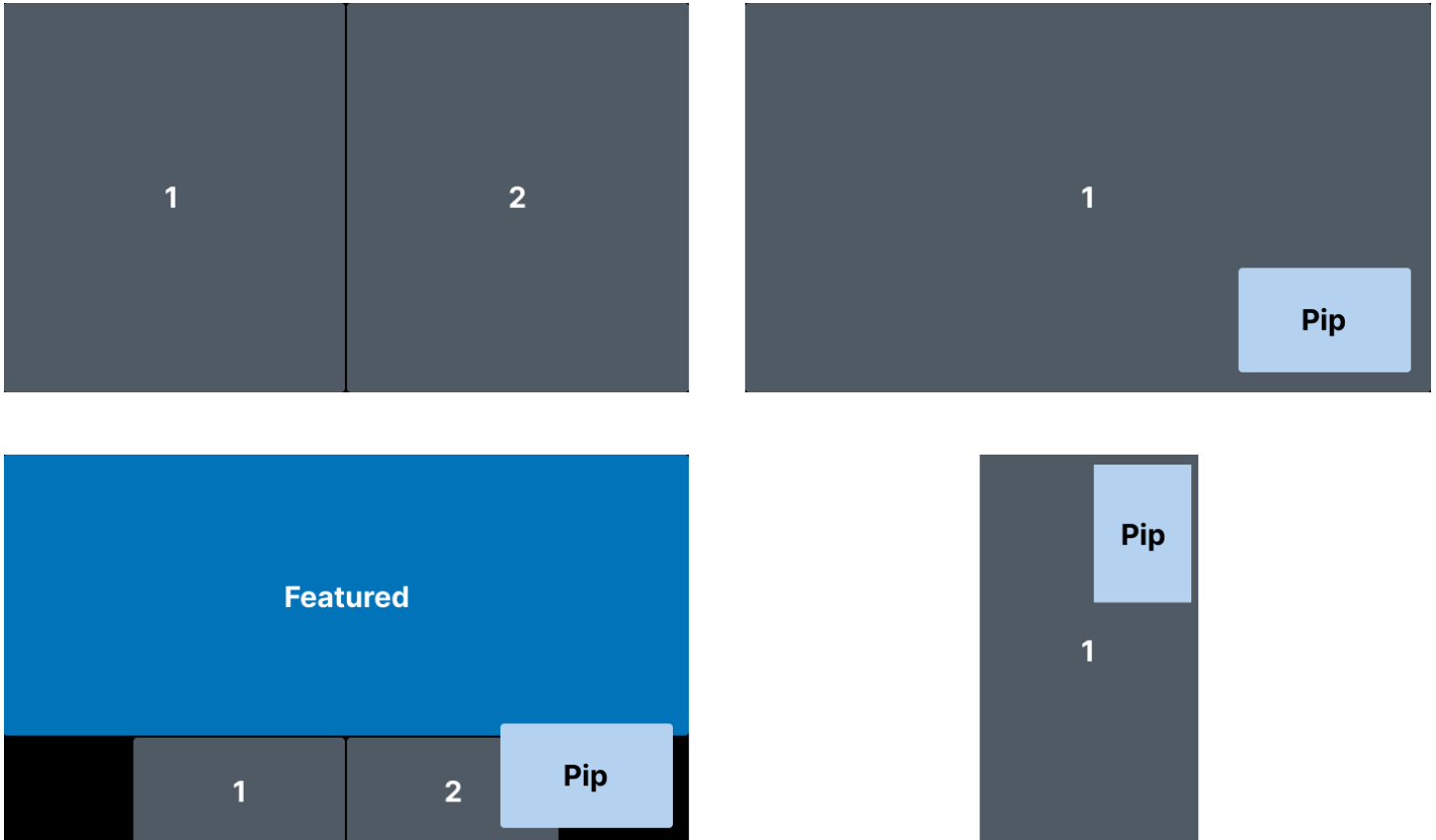
Diseño de imagen en imagen

El diseño de imagen en imagen permite mostrar a un participante en una ventana superpuesta con un tamaño, posición y comportamiento configurables. Entre las propiedades clave se incluyen las siguientes:

- `pipParticipantAttribute` especifica el participante de la ventana de imagen en imagen.
- `pipPosition` determina la posición de la esquina de la ventana de imagen en imagen.
- `pipWidth` y `pipHeight` configuran el ancho y la altura de la ventana de imagen en imagen.
- `pipOffset` establece la posición de desplazamiento de la ventana de imagen en imagen en píxeles desde los bordes más cercanos.
- `pipBehavior` define el comportamiento de imagen en imagen cuando todos los demás participantes se han ido.

Al igual que el diseño de cuadrícula, imagen en imagen admite `featuredParticipantAttribute`, `omitStoppedVideo`, `videoFillMode` y `gridGap` para personalizar aún más la composición.

Para obtener más información sobre el diseño de imagen en imagen (incluidos los valores válidos y los valores predeterminados de todos los campos), consulte el tipo de datos [PipConfiguration](#).



Nota: la resolución máxima admitida por un publicador de escenarios en la composición del servidor es de 1080p. Si un publicador envía un vídeo de más de 1080p, se representará como participante únicamente de audio.

Importante: Asegúrese de que su aplicación no dependa de las características específicas del diseño actual, como el tamaño y la posición de los mosaicos. Se pueden introducir mejoras visuales en los diseños en cualquier momento.

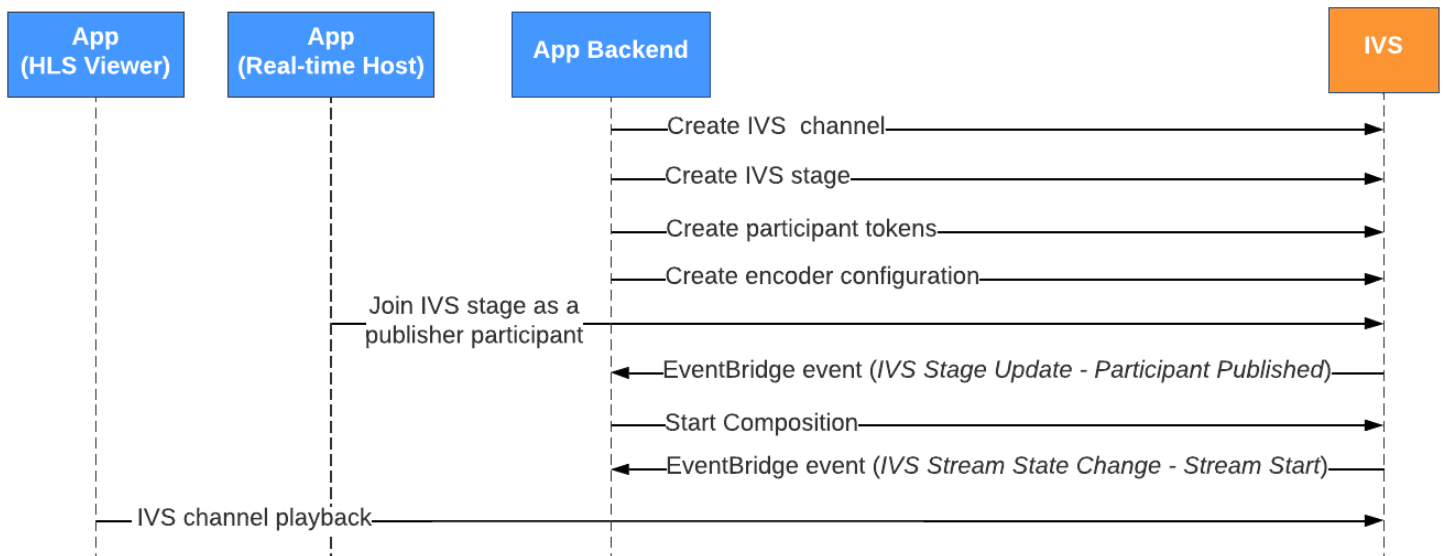
Introducción a la composición del servidor de IVS

Este documento explica los pasos necesarios para comenzar a utilizar la composición del servidor de IVS.

Requisitos previos

Para utilizar la composición del servidor, debe tener un escenario con publicadores activos y utilizar un canal IVS o un bucket de S3 como destino de la composición. A continuación, describimos un posible flujo de trabajo que utiliza los eventos de EventBridge para iniciar una composición que retransmita el escenario a un canal de IVS cuando un participante publique. Como alternativa, puede iniciar y detener las composiciones según la lógica de su propia aplicación. Consulte [Grabación del compuesto](#) para ver otro ejemplo que muestra el uso de la composición del servidor para grabar un escenario directamente en un bucket de S3.

1. Cree un canal IVS. Consulte [Introducción a la transmisión de baja latencia de Amazon IVS](#).
2. Cree un escenario de IVS y tokens de participante para cada publicador.
3. Cree una configuración [EncoderConfiguration](#).
4. Únase al escenario y publique en él. (Consulte las secciones "Publicación y suscripción" de las guías del SDK para transmisiones en tiempo real: [Web](#), [Android](#) e [iOS](#)).
5. Cuando reciba un evento de EventBridge publicado por un participante, llame a [StartComposition](#) con su configuración de diseño deseada.
6. Espere unos segundos y vea la vista compuesta en la reproducción del canal.



Nota: una composición se apaga automáticamente después de 60 segundos de inactividad por parte de los publicadores participantes en el escenario. En ese momento, la composición finaliza y pasa a un estado STOPPED. Una composición se elimina automáticamente después de unos minutos en ese estado STOPPED.

Instrucciones de la CLI

El uso de la CLI de AWS es una opción avanzada y requiere que primero descargue y configure la CLI en su equipo. Para obtener más información, consulte la [Guía del usuario de la interfaz de línea de comandos de AWS](#).

Ahora puede utilizar la CLI para crear y administrar recursos. Los puntos de conexión de la Composición se encuentran debajo del espacio de nombres `ivs-realtime`.

Creación del recurso de EncoderConfiguration

Un EncoderConfiguration es un objeto que permite personalizar el formato del vídeo generado (altura, anchura, velocidad de bits y otros parámetros de transmisión). Puede reutilizar una EncoderConfiguration cada vez que llame al punto de conexión de Composición, como se explica en el siguiente paso.

El siguiente comando crea un recurso EncoderConfiguration que configura los parámetros de composición de vídeo del servidor, como la velocidad de bits del vídeo, la velocidad de fotogramas y la resolución:

```
aws ivs-realtime create-encoder-configuration --name "MyEncoderConfig" --video
  "bitrate=2500000,height=720,width=1280,framerate=30"
```

La respuesta es:

```
{
  "encoderConfiguration": {
    "arn": "arn:aws:ivs:us-east-1:927810967299:encoder-configuration/9W590BY2M8s4",
    "name": "MyEncoderConfig",
    "tags": {},
    "video": {
      "bitrate": 2500000,
      "framerate": 30,
      "height": 720,
      "width": 1280
    }
  }
}
```

Inicialización de una Composición

Con el ARN de EncoderConfiguration proporcionado en la respuesta anterior, cree su recurso de Composición:

Ejemplo de diseño de cuadrícula

```
aws ivs-realtime start-composition --stage-arn "arn:aws:ivs:us-east-1:927810967299:stage/8faHz1SQp0ik" --destinations '[{"channel": {"channelArn": "arn:aws:ivs:us-east-1:927810967299:channel/D01MW4dfMR8r", "encoderConfigurationArn": "arn:aws:ivs:us-east-1:927810967299:encoder-configuration/9W590BY2M8s4"}}]' --layout '{"grid": {"featuredParticipantAttribute": "isFeatured", "videoFillMode": "COVER", "gridGap": 0}}'
```

Ejemplo de diseño de imagen en imagen

```
aws ivs-realtime start-composition --stage-arn "arn:aws:ivs:us-east-1:927810967299:stage/8faHz1SQp0ik" --destinations '[{"channel": {"channelArn": "arn:aws:ivs:us-east-1:927810967299:channel/D01MW4dfMR8r", "encoderConfigurationArn": "arn:aws:ivs:us-east-1:927810967299:encoder-configuration/DEkQHWPVa0w0"}}]' --layout '{"pip": {"pipParticipantAttribute": "isPip", "pipOffset": 10, "pipPosition": "TOP_RIGHT"}}'
```

Nota: Puede utilizar [esta herramienta](#) para generar más fácilmente la configuración de `--layout` en función de sus elecciones de diseño.

La respuesta mostrará que la Composición se creó con un estado `STARTING`. Una vez que la Composición comience a publicarse, el estado pasará a `ACTIVE`. (Para ver el estado, llame al punto de conexión `ListCompositions` o `GetComposition`).

Una vez que una Composición sea `ACTIVE`, la vista compuesta del escenario IVS será visible en el canal IVS mediante `ListCompositions`:

```
aws ivs-realtime list-compositions
```

La respuesta es:

```
{
  "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": {}  
}  
]  
}
```

Nota: es necesario que los publicadores participantes publiquen activamente en el escenario para mantener viva la composición. Para obtener más información, consulte las secciones "Publicaciones y suscripciones" de las guías del SDK para transmisiones en tiempo real: [Web](#), [Android](#) e [iOS](#). Debe crear un token de escenario distinto para cada participante.

Habilitación de la pantalla compartida en la composición del servidor de IVS

Para usar un diseño de pantalla compartida fijo, siga los pasos que se indican a continuación.

Creación del recurso de EncoderConfiguration

El siguiente comando crea un recurso EncoderConfiguration que configura los parámetros de composición del servidor (velocidad de bits, velocidad de fotogramas y resolución de vídeo).

```
aws ivs-realtime create-encoder-configuration --name "test-ssc-with-screen-share" --  
video={bitrate=2000000, framerate=30, height=720, width=1280}
```

Cree un token de participante en el escenario con un atributo screen-share. Como especificaremos screen-share como el nombre del espacio featured, necesitamos crear un token de escenario con el atributo screen-share establecido en true:


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



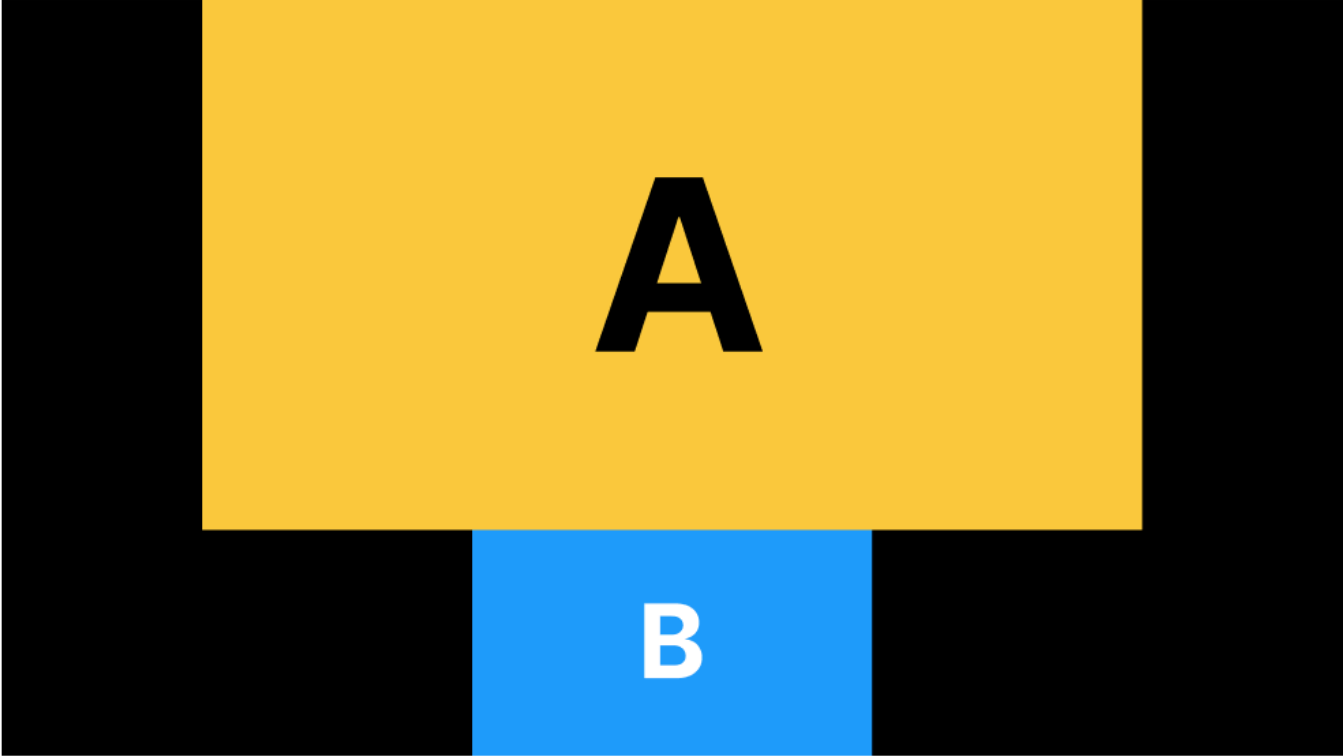
```
"grid" : {
  "featuredParticipantAttribute" : "screen-share",
  "gridGap": 2,
  "omitStoppedVideo": false,
  "videoAspectRatio": "VIDEO"
},
"stageArn" : "arn:aws:ivs:us-east-1:927810967299:stage/8faHz1SQp0ik",
"startTime" : "2023-09-27T21:32:38Z",
"state" : "STARTING",
"tags" : { }
}
```

Cuando el participante del escenario E813MFk1PWLF se una al escenario, el vídeo de ese participante se mostrará en el espacio destacado y todos los demás publicadores del escenario se renderizarán debajo del espacio:

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

Detención de la composición

Para detener una composición en cualquier momento, llame al punto de conexión `StopComposition`:

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


Grabación de IVS | Transmisión en tiempo real

Existen dos opciones de grabación para Transmisión en tiempo real de IVS:

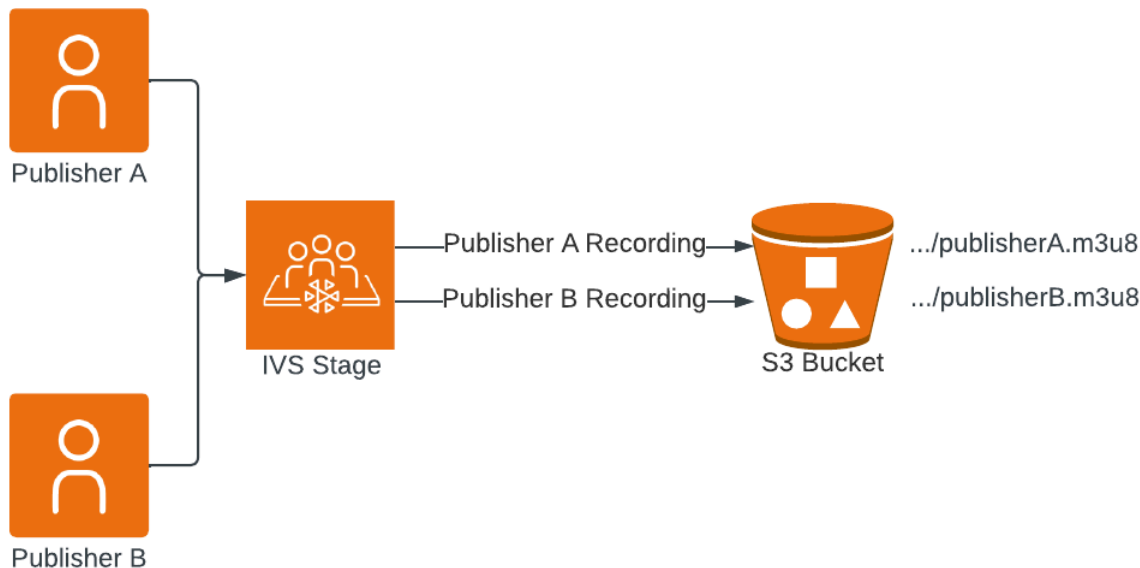
- Con la grabación de participantes individuales, el contenido multimedia de cada publicador se graba en archivos independientes.
- Por el contrario, la grabación compuesta combina contenido multimedia de todos los publicadores en una sola vista y lo graba en un solo archivo.

La grabación de participantes individuales no conlleva cargos adicionales de Amazon IVS, mientras que la grabación compuesta conlleva cargos según la tarifa por hora del video codificado. Ambas opciones de grabación conllevan los cargos estándar de almacenamiento y solicitud de S3. Para obtener más información, consulte [Precios de Amazon IVS](#).

Si busca una solución más personalizable, considere utilizar el proyecto [IVSStageSaver](#) de código abierto como base para su propio servicio de grabación alojado por cuenta propia.

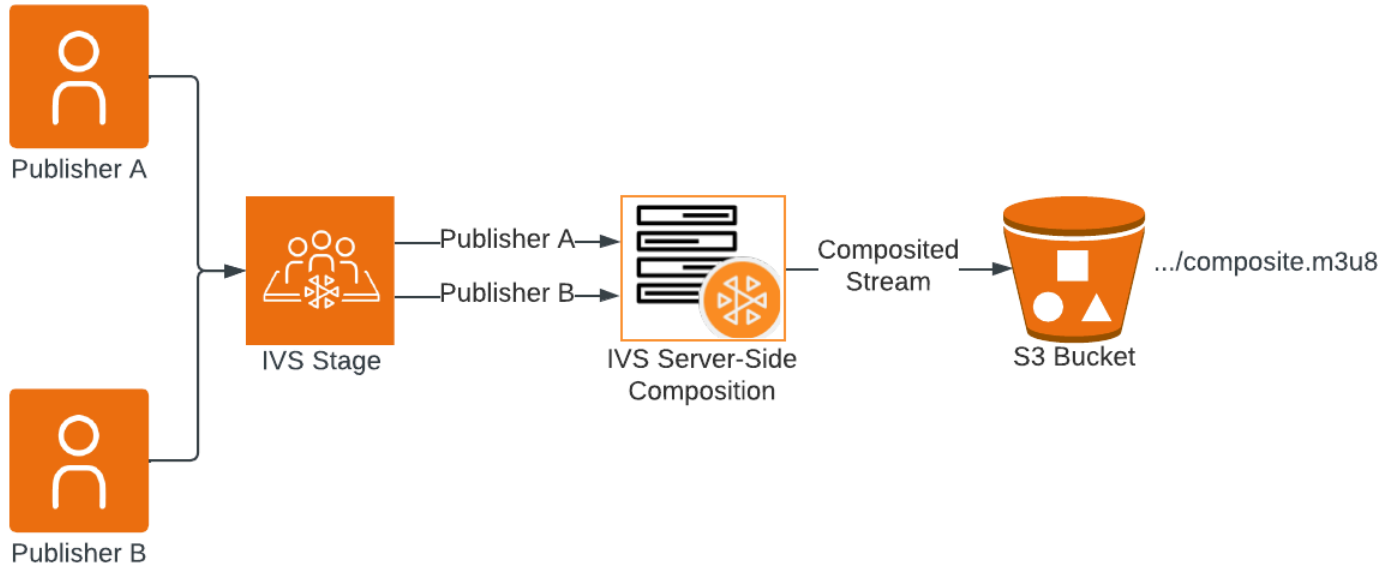
Grabación de participantes individuales

Esta opción es ideal para transmisiones en directo con un solo publicador o cuando se necesitan grabaciones independientes de cada publicador, especialmente con fines de moderación. Para obtener más información, consulte [Grabación de participantes individuales](#).



Grabación compuesta

Esta opción combina contenido multimedia de varios publicadores en una sola vista y lo graba en un solo archivo, lo que resulta ideal para disfrutar de una experiencia de video bajo demanda. Para obtener más información, consulte [Grabación compuesta](#).



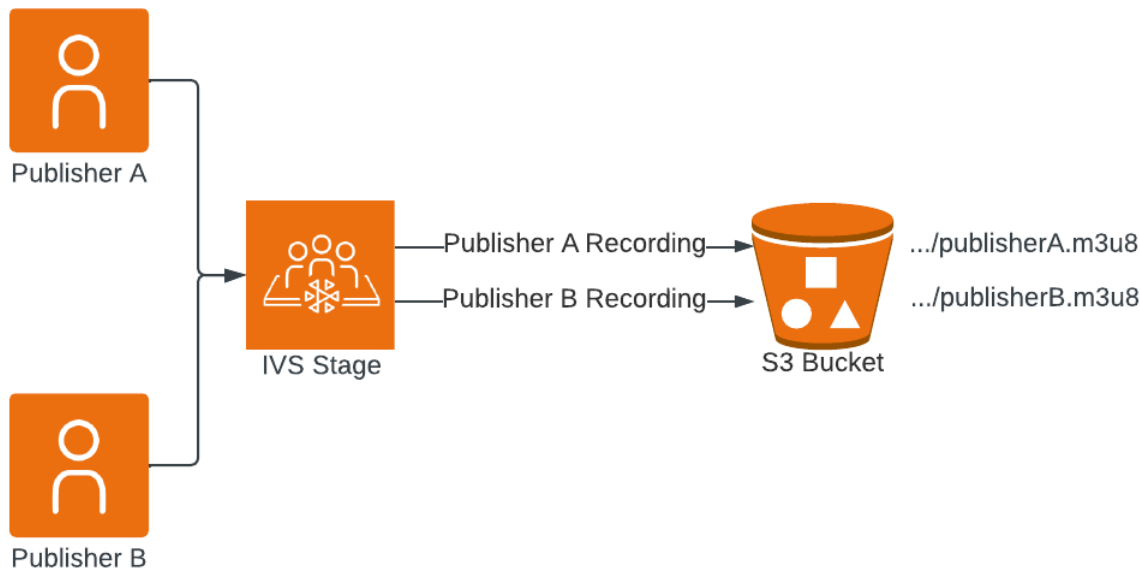
Grabación de participantes individuales en IVS | Transmisión en tiempo real

Este documento explica cómo utilizar la grabación de participantes individuales con la transmisión en tiempo real de IVS.

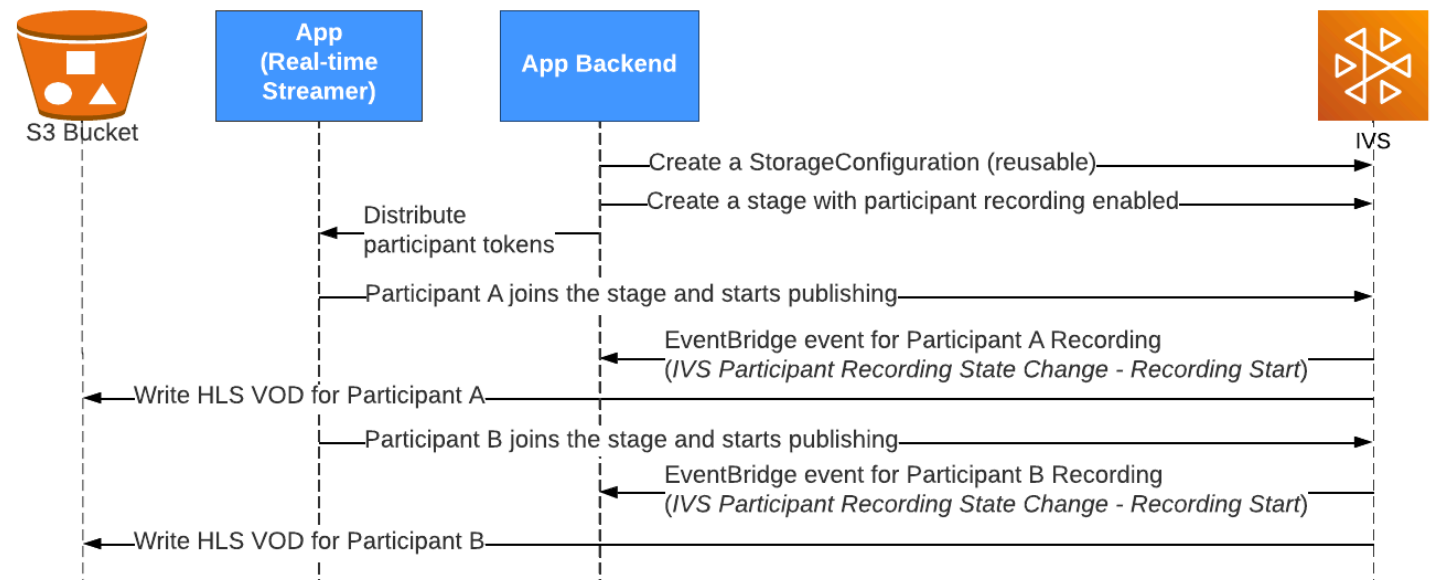
Introducción

La grabación de participantes individuales permite a los clientes de IVS que transmiten en tiempo real grabar a los publicadores de fases de IVS de forma individual en buckets de S3. Cuando una fase permite la grabación de participantes individuales, el contenido del publicador se graba cuando este comienza a publicar en la fase.

Nota: Si necesita mezclar a todos los participantes de la fase en un solo video, la característica de grabación compuesta es la más adecuada. Consulte [Grabación](#) para ver un resumen de la grabación de contenido de transmisión en tiempo real de IVS.



Flujo de trabajo



1. Creación de S3 Bucket

Necesitará un bucket de S3 para escribir VOD. Para obtener más información, consulte la documentación de S3 sobre [cómo crear buckets](#). Tenga en cuenta que, para la grabación de participantes individuales, los buckets de S3 deben crearse en la misma región de AWS que la fase de IVS.

Importante: Si utiliza un bucket de S3 existente, la configuración Propiedad del objeto debe ser Aplicada al propietario del bucket o Propietario del bucket preferido. Para obtener más información, consulte la documentación de S3 sobre el [control de la propiedad de los objetos](#).

2. Creación de un objeto StorageConfiguration

Tras crear un bucket, llame a la API de transmisión en tiempo real de IVS para [crear un objeto StorageConfiguration](#). Una vez que la configuración de almacenamiento se haya creado correctamente, IVS tendrá permiso para escribir en el bucket de S3 proporcionado. Puede reutilizar este objeto StorageConfiguration en varias fases.

3. Creación de una fase con tokens de participantes

Ahora tiene que [crear una fase de IVS](#) con la grabación de participantes individuales habilitada (configurando el objeto AutoParticipantRecordingConfiguration), así como tokens de participantes para cada publicador.

La siguiente solicitud crea una fase con dos tokens de participantes y con la grabación de participantes individuales habilitada.

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

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

4. Unión a la fase como publicador activo

Distribuya los tokens de participantes entre los publicadores y haga que se unan a la fase y comiencen [a publicar en ella](#).

Cuando se unan a la fase y comienzan a publicar en ella mediante uno de los [SDK de transmisión en tiempo real de IVS](#), el proceso de grabación de los participantes se inicia automáticamente y le envía un [evento de EventBridge](#) en el que se indica que la grabación ha comenzado. (El evento es Cambio en el estado de la grabación del participante de IVS: inicio de la grabación). Al mismo tiempo, el proceso de grabación de los participantes comienza a escribir los archivos de VOD y de metadatos en el bucket de S3 configurado. Nota: No se garantiza que los participantes conectados durante periodos extremadamente cortos (menos de 5 segundos) se graben.

Existen dos formas de obtener el prefijo de S3 para cada grabación:

- Escuche el evento de EventBridge:

```
{
  "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"
  }
}
```

- Utilice el punto de conexión de la API [GetParticipant](#): la respuesta incluye el bucket de S3 y el prefijo del lugar donde se graba al participante. La solicitud es la siguiente:

```
POST /GetParticipant HTTP/1.1
Content-type: application/json
{
```

```
"participantID": "xYz1c2d3e4f",
"sessionId": "st-ZyXwvu1T2s",
"stageArn": "arn:aws:ivs:us-west-2:123456789012:stage/AbCdef1G2hij"
}
```

Y esta es la respuesta:

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

5. Reproducción del VOD

Una vez finalizada la grabación, puede verla con el [reproductor de IVS](#). Consulte [Playback of Recorded Content from Private Buckets](#) para obtener instrucciones sobre cómo configurar las distribuciones de CloudFront para la reproducción de VOD.

Grabación solo de audio

Al configurar la grabación de un participante individual, puede elegir que solo se escriban segmentos de audio HLS en su bucket de S3. Para utilizar esta característica, elija `AUDIO_ONLY` `mediaType` al crear la fase:

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

{
  "autoParticipantRecordingConfiguration": {
    "storageConfigurationArn": "arn:aws:ivs:us-west-2:123456789012:storage-configuration/AbCdef1G2hij",
    "mediaTypes": ["AUDIO_ONLY"]
  },
  "name": "TestStage",
  "participantTokenConfigurations": [
```

```
{
  "capabilities": ["PUBLISH", "SUBSCRIBE"],
  "duration": 20160,
  "userId": "1"
},
{
  "capabilities": ["PUBLISH", "SUBSCRIBE"],
  "duration": 20160,
  "userId": "2"
}
]
```

Grabación de contenidos

Cuando la grabación de un participante individual esté activa, empezará a ver cómo se escriben segmentos de video HLS y archivos de metadatos en el bucket de S3 que se proporcionó al crear la fase. Estos contenidos están disponibles para el procesamiento posterior o la reproducción como video bajo demanda.

Tenga en cuenta que, una vez finalizada la grabación, se envía un evento Cambio en el estado de la grabación del participante de IVS: fin de la grabación a través de EventBridge. Le recomendamos que reproduzca o procese transmisiones grabadas solo después de que se reciba este evento. Para obtener más detalles, vea [Uso de EventBridge con la transmisión en tiempo real de IVS](#).

A continuación, se muestra una estructura de directorios de ejemplo y el contenido de una grabación de una sesión en vivo de IVS:

```
s3://mybucket/stageId/stageSessionId/participantId/timestamp
  events
    recording-started.json
    recording-ended.json
  media
    hls
  multivariant.m3u8
    high
      playlist.m3u8
      1.mp4
```

La carpeta `events` contiene los archivos de metadatos correspondientes al evento de grabación. Los archivos de metadatos JSON se generan cuando la grabación se inicia, finaliza correctamente o termina con errores:

- `events/recording-started.json`
- `events/recording-ended.json`
- `events/recording-failed.json`

Una carpeta `events` contiene `recording-started.json` y `recording-ended.json` o `recording-failed.json`. Estos contienen metadatos relacionados con la sesión grabada y sus formatos de salida. Los detalles de JSON se dan a continuación.

La carpeta `media` contiene el contenido multimedia compatible. La subcarpeta `hls` contiene todos los archivos multimedia y de manifiesto generados durante la sesión de grabación y se puede reproducir con el reproductor de IVS.

Archivos de metadatos JSON

Estos metadatos están en formato JSON. Contiene la siguiente información:

Campo	Tipo	Obligatorio	Descripción
<code>stage_arn</code>	string	Sí	ARN de la fase que se utiliza como origen de la grabación.
<code>session_id</code>	cadena	Sí	Cadena que representa el valor de <code>session_id</code> de la fase en la que se graba al participante.
<code>participant_id</code>	cadena	Sí	Cadena que representa el identificador del participante grabado.
<code>recording_started_at</code>	cadena	Condicional	Marca de hora UTC RFC 3339 cuando se inició la grabación. No está disponible cuando <code>recording_status</code> está <code>RECORDING_START_FA</code>

Campo	Tipo	Obligatorio	Descripción
			ILED . Consulte también la nota siguiente para obtener más información sobre recording_ended_at .
recording_ended_at	cadena	Condicional	<p>Marca de tiempo UTC RFC 3339 cuando finalizó la grabación . Esto solo está disponible cuando recording_status es "RECORDING_ENDED" o "RECORDING_ENDED_WITH_FAILURE" .</p> <p>Nota: recording_started_at y recording_ended_at son marcas de tiempo que indican cuando se generan estos eventos y es posible que no coincidan exactamente con las marcas de tiempo del segmento de video HLS. Para determinar con precisión la duración de una grabación, utilice la duration_ms .</p>
recording_status	cadena	Sí	El estado de la grabación. Valores válidos: "RECORDING_STARTED" , "RECORDING_ENDED" , "RECORDING_START_FAILED" , "RECORDING_ENDED_WITH_FAILURE" .

Campo	Tipo	Obligatorio	Descripción
<code>recording_status_message</code>	cadena	Condicional	Información descriptiva sobre el estado. Esto solo está disponible cuando <code>recording_status</code> es "RECORDING_ENDED" o "RECORDING_ENDED_WITH_FAILURE".
<code>media</code>	objeto	Sí	Objeto que contiene los objetos enumerados de contenido multimedia disponibles para esta grabación. Valor válido: "hls".
<code>hls</code>	objeto	Sí	Campo enumerado que describe la salida del formato HLS de Apple.
<code>duration_ms</code>	integer	Condicional	Duración del contenido HLS grabado en milisegundos. Esto solo está disponible cuando <code>recording_status</code> es "RECORDING_ENDED" o "RECORDING_ENDED_WITH_FAILURE". Si se produjo un error antes de realizar cualquier grabación, esto es 0.
<code>path</code>	string	Sí	Ruta relativa del prefijo S3 donde se almacena el contenido HLS.
<code>playlist</code>	string	Sí	Nombre del archivo de lista de reproducción maestra HLS.

Campo	Tipo	Obligatorio	Descripción
renditions	objeto	Sí	Matriz de representaciones (variantes HLS) de objetos de metadatos. Siempre hay al menos una copia.
path	string	Sí	Ruta relativa del prefijo S3 donde se almacena el contenido HLS para esta copia.
playlist	string	Sí	Nombre del archivo de lista de reproducción multimedia para esta copia.
version	cadena	Sí	La versión del esquema de metadatos.

Ejemplo: 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"
        }
      ]
    }
  }
}
```

```
}
```

Ejemplo: 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"
        }
      ]
    }
  }
}
```

Ejemplo: recording-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"  
      }  
    ]  
  }  
}
```

Grabación compuesta de IVS | Transmisión en tiempo real

Este documento explica cómo utilizar la característica de grabación compuesta en la [composición del servidor](#). La grabación compuesta permite generar grabaciones HLS de un escenario IVS al combinar de forma eficaz todos los publicadores de escenarios en una sola vista mediante un servidor IVS y, a continuación, guardar el vídeo resultante en un bucket de S3.

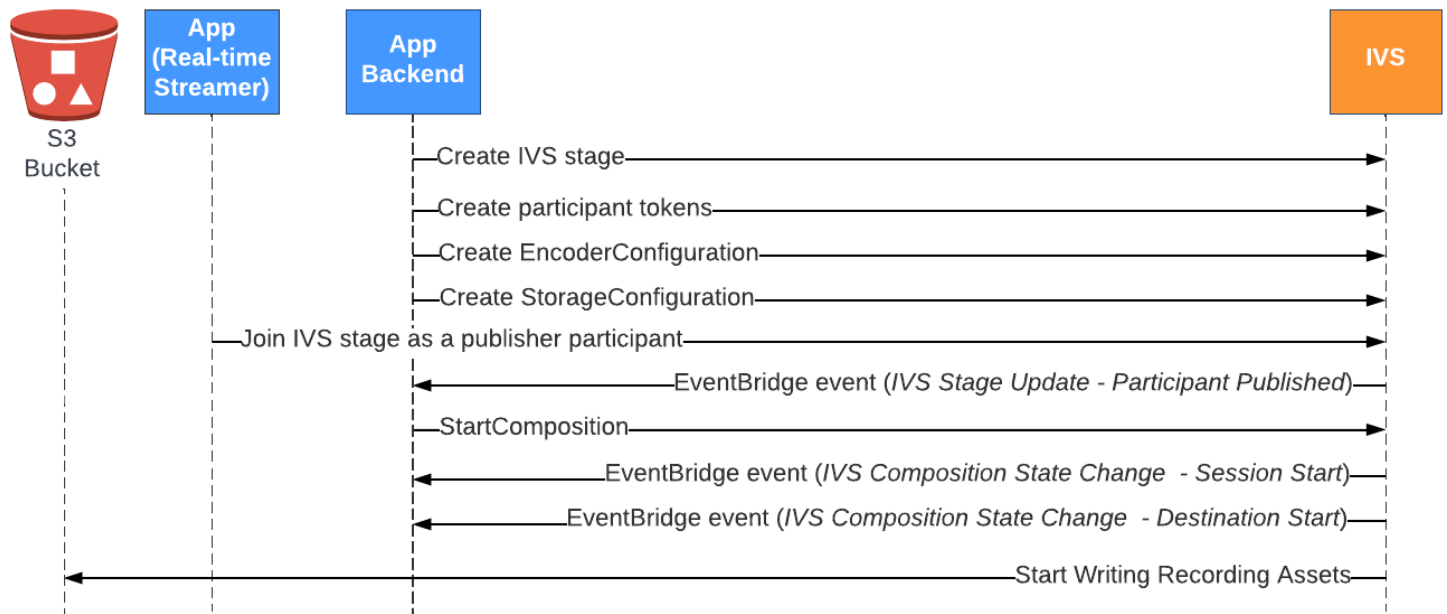
Requisitos previos

Para utilizar la grabación compuesta, debe disponer de una fase con publicadores activos y un bucket de S3 para utilizarlo como destino de grabación. A continuación, describimos un posible flujo de trabajo que utiliza eventos de EventBridge para grabar una composición en un bucket de S3. Como alternativa, puede iniciar y detener las composiciones según la lógica de su propia aplicación.

1. Cree [un escenario de IVS](#) y tokens de participante para cada publicador.
2. Cree una [EncoderConfiguration](#) (un objeto que represente cómo debe renderizarse el vídeo grabado).
3. Cree un [bucket de S3](#) y una [StorageConfiguration](#) (donde se almacenará el contenido de la grabación).

Importante: Si utiliza un bucket de S3 existente, la configuración Propiedad del objeto debe ser Aplicada al propietario del bucket o Propietario del bucket preferido. Para obtener más información, consulte la documentación de S3 sobre el [control de la propiedad de los objetos](#).

4. [Únase al escenario y publique en él](#).
5. Cuando reciba un [evento de EventBridge](#) publicado por un participante, llame a [startComposition](#) con un objeto DestinationConfiguration de S3 como destino.
6. Tras unos segundos, debería poder ver cómo los segmentos HLS se conservan en sus buckets de S3.



Nota: Una composición se apaga automáticamente después de 60 segundos de inactividad por parte de los publicadores participantes en el escenario. En ese momento, la composición finaliza y pasa a un estado STOPPED. Una composición se borra automáticamente después de unos minutos en ese estado STOPPED. Para obtener más información, consulte [Ciclo de vida de la composición](#) en Composición del servidor.

Ejemplo de grabación compuesta: inicie la composición con un destino de bucket de S3

El siguiente ejemplo muestra una llamada típica al punto de conexión [StartComposition](#), especificando S3 como el único destino de la composición. Una vez que la composición pase a un estado ACTIVE, los segmentos de vídeo y los metadatos comenzarán a escribirse en el bucket de S3 especificado por el objeto `storageConfiguration`. Para crear composiciones con diferentes diseños, consulte la sección “Diseños” en [Composición del servidor](#) y la [Referencia de la API de transmisión en tiempo real de IVS](#).

Solicitud

```

POST /StartComposition HTTP/1.1
Content-type: application/json

{
  "destinations": [
    {
      "s3": {

```

```

    "encoderConfigurationArns": [
      "arn:aws:ivs:ap-northeast-1:927810967299:encoder-configuration/
PAAwglkRtjge"
    ],
    "storageConfigurationArn": "arn:aws:ivs:ap-
northeast-1:927810967299:storage-configuration/ZBcEbgE24Cq"
  }
},
"idempotencyToken": "db1i782f1g9",
"stageArn": "arn:aws:ivs:ap-northeast-1:927810967299:stage/WyGkzNFGwiwr"
}

```

Respuesta

```

{
  "composition": {
    "arn": "arn:aws:ivs:ap-northeast-1:927810967299:composition/s2AdaGUbvQgp",
    "destinations": [
      {
        "configuration": {
          "name": "",
          "s3": {
            "encoderConfigurationArns": [
              "arn:aws:ivs:ap-northeast-1:927810967299:encoder-
configuration/PAAwglkRtjge"
            ],
            "recordingConfiguration": {
              "format": "HLS"
            },
            "storageConfigurationArn": "arn:aws:ivs:ap-
northeast-1:927810967299:storage-configuration/ZBcEbgE24Cq"
          }
        },
        "detail": {
          "s3": {
            "recordingPrefix": "MNALAch9j2EJ/s2AdaGUbvQgp/2pBRKrNgX1ff/
composite"
          }
        },
        "id": "2pBRKrNgX1ff",
        "state": "STARTING"
      }
    ]
  }
}

```

```

    ],
    "layout": null,
    "stageArn": "arn:aws:ivs:ap-northeast-1:927810967299:stage/WyGkzNFGwiwr",
    "startTime": "2023-11-01T06:25:37Z",
    "state": "STARTING",
    "tags": {}
  }
}

```

El campo `recordingPrefix`, presente en la respuesta de `StartComposition`, se puede utilizar para determinar dónde se almacenará el contenido de la grabación.

Grabación de contenidos

Cuando la composición pase a un estado `ACTIVE`, empezará a ver cómo se escriben segmentos de vídeo HLS y archivos de metadatos en el bucket de S3 que se proporcionó al llamar a `startComposition`. Estos contenidos están disponibles para el procesamiento posterior o la reproducción como vídeo bajo demanda.

Tenga en cuenta que una vez que se publica una composición, se emite un evento de “Cambio en el estado de la composición de IVS” y se puede tardar un poco en escribir los archivos de manifiesto y los segmentos de vídeo. Le recomendamos que reproduzca o procese transmisiones grabadas solo después de que se reciba el evento “Cambio en el estado de la composición de IVS (fin de la sesión)” Para obtener más detalles, vea [Uso de EventBridge con la transmisión en tiempo real de IVS](#).

A continuación, se muestra una estructura de directorios de ejemplo y el contenido de una grabación de una sesión en vivo de IVS:

```

MNALAch9j2EJ/s2AdaGubvQgp/2pBRK1NgX1ff/composite
  events
    recording-started.json
    recording-ended.json
  media
    hls

```

La carpeta `events` contiene los archivos de metadatos correspondientes al evento de grabación. Los archivos de metadatos JSON se generan cuando la grabación se inicia, finaliza correctamente o termina con errores:

- `events/recording-started.json`

- `events/recording-ended.json`
- `events/recording-failed.json`

Una carpeta `events` contendrá `recording-started.json` y `recording-ended.json` o `recording-failed.json`.

Estos contienen metadatos relacionados con la sesión grabada y sus formatos de salida. Los detalles de JSON se dan a continuación.

La carpeta `media` contiene el contenido multimedia compatible. La subcarpeta `hls` contiene todos los archivos multimedia y de manifiesto generados durante la sesión compuesta y se puede reproducir con el reproductor de IVS. El manifiesto HLS se encuentra en la carpeta `multivariant.m3u8`.

Política de bucket para StorageConfiguration

Cuando se crea un objeto `StorageConfiguration`, IVS tendrá acceso para escribir contenido en el bucket de S3 especificado. Este acceso se concede mediante modificaciones en la política del bucket de S3. Si la política del bucket se modifica de forma que se elimine el acceso del IVS, las grabaciones nuevas y en curso fallarán.

El siguiente ejemplo muestra una política de bucket de S3 que permite a IVS escribir en el bucket de S3:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CompositeWrite-y1d212y",
      "Effect": "Allow",
      "Principal": {
        "Service": "ivs-composite.ap-northeast-1.amazonaws.com"
      },
      "Action": [
        "s3:PutObject",
        "s3:PutObjectAcl"
      ],
      "Resource": "arn:aws:s3:::my-s3-bucket/*",
      "Condition": {
        "StringEquals": {
```

```

        "s3:x-amz-acl": "bucket-owner-full-control"
    },
    "Bool": {
        "aws:SecureTransport": "true"
    }
}
]
}

```

Archivos de metadatos JSON

Estos metadatos están en formato JSON. Contiene la siguiente información:

Campo	Tipo	Obligatorio	Descripción
stage_arn	string	Sí	ARN del escenario que se utiliza como fuente de la composición.
media	objeto	Sí	Objeto que contiene los objetos enumerados de contenido multimedia disponibles para esta grabación. Valores válidos: "hls".
hls	objeto	Sí	Campo enumerado que describe la salida del formato HLS de Apple.
duration_ms	integer	Condicional	Duración del contenido HLS grabado en milisegundos. Esto solo está disponible cuando <code>recording_status</code> es "RECORDING_ENDED" o "RECORDING_ENDED_WITH_FAILURE". Si se produjo un error antes de realizar cualquier grabación, esto es 0.

Campo	Tipo	Obligatorio	Descripción
path	string	Sí	Ruta relativa del prefijo S3 donde se almacena el contenido HLS.
playlist	string	Sí	Nombre del archivo de lista de reproducción maestra HLS.
renditions	objeto	Sí	Matriz de copias (variante HLS) de objetos de metadatos. Siempre hay al menos una copia.
path	string	Sí	Ruta relativa del prefijo S3 donde se almacena el contenido HLS para esta copia.
playlist	string	Sí	Nombre del archivo de lista de reproducción multimedia para esta copia.
resolution_height	int	Condicional	Altura de resolución de píxeles del video codificado. Solo está disponible cuando la copia contiene una pista de video.
resolution_width	int	Condicional	Ancho de resolución de píxeles del video codificado. Solo está disponible cuando la copia contiene una pista de video.

Campo	Tipo	Obligatorio	Descripción
<code>recording_ended_at</code>	cadena	Condicional	<p>Marca de tiempo UTC RFC 3339 cuando finalizó la grabación. Esto solo está disponible cuando <code>recording_status</code> es <code>"RECORDING_ENDED"</code> o <code>"RECORDING_ENDED_WITH_FAILURE"</code>.</p> <p><code>recording_started_at</code> y <code>recording_ended_at</code> son marcas de tiempo cuando se generan estos eventos y es posible que no coincidan exactamente con las marcas de tiempo del segmento de video HLS. Para determinar con precisión la duración de una grabación, utilice la <code>duration_ms</code>.</p>
<code>recording_started_at</code>	cadena	Condicional	<p>Marca de hora UTC RFC 3339 cuando se inició la grabación. No está disponible cuando <code>recording_status</code> está <code>RECORDING_START_FAILED</code>.</p> <p>Consulte la nota que figura arriba para <code>recording_ended_at</code>.</p>

Campo	Tipo	Obligatorio	Descripción
<code>recording_status</code>	cadena	Sí	El estado de la grabación. Valores válidos: "RECORDING_STARTED", "RECORDING_ENDED", "RECORDING_START_FAILED", "RECORDING_ENDED_WITH_FAILURE".
<code>recording_status_message</code>	cadena	Condicional	Información descriptiva sobre el estado. Esto solo está disponible cuando <code>recording_status</code> es "RECORDING_ENDED" o "RECORDING_ENDED_WITH_FAILURE".
<code>version</code>	cadena	Sí	La versión del esquema de metadatos.

Ejemplo: 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
        }
      ]
    }
  ]
}
```

```
    }  
  }  
}
```

Ejemplo: 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  
        }  
      ]  
    }  
  }  
}
```

Ejemplo: recording-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  
        }  
      ]  
    }  
  }  
}
```

```
    "renditions": [  
      {  
        "path": "720p30-abcdeABCDE12",  
        "playlist": "playlist.m3u8",  
        "resolution_width": 1280,  
        "resolution_height": 720  
      }  
    ]  
  }  
}
```

Reproducción de contenido grabado desde buckets privados

El contenido grabado es privado de forma predeterminada; por lo tanto, no se puede acceder a ellos para reproducirlos mediante la dirección URL directa de S3. Si intenta abrir la lista de reproducción multivariada HLS (archivo m3u8) para su reproducción mediante el reproductor de IVS u otro reproductor, recibirá un error (por ejemplo, “You do not have permission to access the requested resource” [No tiene permiso para acceder al recurso solicitado]). En su lugar, puede reproducir estos archivos mediante la Red de entrega de contenido (CDN, Content Delivery Network) de Amazon CloudFront.

Las distribuciones de CloudFront pueden configurarse para distribuir contenido desde buckets privados. Por lo general, esto es preferible a tener buckets de acceso abierto donde las lecturas omiten los controles que ofrece CloudFront. Puede configurar la distribución para que funcione desde un bucket privado mediante la creación de un control de acceso de origen (OAC), que es un usuario especial de CloudFront que tiene permisos de lectura en el bucket de origen privado. Puede crear el OAC cuando cree su distribución a través de la consola o la API de CloudFront. Consulte [Creación de un nuevo control de acceso de origen](#) en la Guía para desarrolladores de Amazon CloudFront.

Configuración de la reproducción mediante CloudFront con CORS activado

En este ejemplo, se explica cómo un desarrollador puede configurar una distribución de CloudFront con CORS habilitado, lo que permite la reproducción de sus grabaciones desde cualquier dominio. Esto resulta especialmente útil durante la fase de desarrollo, pero puede modificar el siguiente ejemplo para adaptarlo a sus necesidades de producción.

Paso 1: crear un bucket de S3

Crear un bucket de S3 que se utilizará para almacenar las grabaciones. Tenga en cuenta que el bucket debe estar en la misma región que utiliza para su flujo de trabajo de IVS.

Agregar una política de permisos CORS al bucket:

1. En la consola de AWS, vaya a la pestaña Permisos de bucket de S3.
2. Copie la política de CORS que se encuentra a continuación y péguela en Cross-origin resource sharing (CORS). Esto habilitará el acceso a CORS en el bucket de S3.

```
[
  {
    "AllowedHeaders": [
      "*"
    ],
    "AllowedMethods": [
      "PUT",
      "POST",
      "DELETE",
      "GET"
    ],
    "AllowedOrigins": [
      "*"
    ],
    "ExposeHeaders": [
      "x-amz-server-side-encryption",
      "x-amz-request-id",
      "x-amz-id-2"
    ]
  }
]
```

Paso 2: crear una distribución de CloudFront.

Consulte [Creación de una distribución de CloudFront](#) en la Guía para desarrolladores de CloudFront.

Mediante la consola de AWS, ingrese la siguiente información:

Para este campo...	Elija esto...
Dominio de origen	El bucket de S3 en el paso anterior.

Para este campo...	Elija esto...
Acceso de origen	Configuración del control de acceso de origen (recomendado), mediante parámetros por defecto
Comportamiento predeterminado de la caché: política de protocolo de visualización	Redireccionamiento de HTTP a HTTPS
Comportamiento predeterminado de la caché: métodos HTTP permitidos	GET, HEAD y OPTIONS
Comportamiento predeterminado de la caché: solicitudes de origen y clave de caché	Política de caché deshabilitada
Comportamiento de caché predeterminado: política de solicitudes de Origen	CORS-S3Origin
Comportamiento predeterminado de la caché: política de encabezados de respuesta	SimpleCORS
firewall de aplicaciones web	Habilitar protecciones de seguridad

A continuación, guarde la distribución de CloudFront.

Paso 3: configurar la política de bucket de S3

1. Elimine cualquier configuración de almacenamiento que haya configurado para el bucket de S3. Esto eliminará todas las políticas de bucket que se hayan agregado automáticamente al crear la política para ese bucket.
2. Vaya a su distribución de CloudFront, asegúrese de que todos los campos de distribución estén en los estados definidos en el paso anterior y copie la política de bucket (utilice el botón Copiar política).
3. Vaya a su bucket de S3. En la pestaña Permisos, seleccione Editar política de bucket y pegue la política de bucket que copió en el paso anterior. Tras este paso, la política de bucket debe tener exclusivamente la política de CloudFront.
4. Cree una StorageConfiguration, especificando el bucket de S3.

Una vez creada la StorageConfiguration, verá dos elementos en la política de bucket de S3: uno que permite a CloudFront leer el contenido y otro que permite a IVS escribir contenido. Un ejemplo de política de bucket final, con acceso a CloudFront e IVS, se muestra en [Ejemplo: Política de bucket de S3 con CloudFront e IVS Access](#).

Paso 4: reproducir grabaciones

Tras configurar correctamente la distribución de CloudFront y actualizar la política de bucket, debería poder reproducir las grabaciones con el reproductor IVS:

1. Inicie correctamente una composición y asegúrese de tener una grabación almacenada en el bucket de S3.
2. Tras seguir los pasos 1 a 3 de este ejemplo, los archivos de vídeo deben estar disponibles para su utilización a través de la URL de CloudFront. Su URL de CloudFront es el Nombre del dominio de distribución en la pestaña Detalles de la consola de Amazon CloudFront. Debe tener un aspecto similar al siguiente:

```
a1b23cdef4ghij.cloudfront.net
```

3. Para reproducir el video grabado a través de la distribución de CloudFront, busque la clave de objeto para su archivo `multivariant.m3u8` en el bucket de s3. Debe tener un aspecto similar al siguiente:

```
FDew6Szq5iTT/9NIpWJHj0wPT/fjFKbylPb3k4/composite/media/hls/  
multivariant.m3u8
```

4. Agregue la clave de objeto al final de la URL de CloudFront. Su URL final tendrá un aspecto similar al siguiente:

```
https://a1b23cdef4ghij.cloudfront.net/FDew6Szq5iTT/9NIpWJHj0wPT/  
fjFKbylPb3k4/composite/media/hls/multivariant.m3u8
```

5. Ahora puede añadir la URL final al atributo fuente de un reproductor IVS para ver la grabación completa. Para ver el vídeo grabado, puede utilizar la demostración en [Cómo empezar](#) en el SDK del reproductor de IVS: Guía web.

Ejemplo: política de bucket de S3 con CloudFront e IVS Access

El siguiente fragmento ilustra una política de bucket de S3 que permite a CloudFront leer contenido en el bucket privado e IVS para escribir contenido en el bucket. Nota: No copie ni pegue el siguiente

fragmento en su propio bucket. Su política debe contener los ID relevantes para su distribución de CloudFront y StorageConfiguration.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CompositeWrite-7eiKaIGkC9D0",
      "Effect": "Allow",
      "Principal": {
        "Service": "ivs-composite.ap-northeast-1.amazonaws.com"
      },
      "Action": [
        "s3:PutObject",
        "s3:PutObjectAcl"
      ],
      "Resource": "arn:aws:s3:::eicheane-test-1026-2-ivs-recordings/*",
      "Condition": {
        "StringEquals": {
          "s3:x-amz-acl": "bucket-owner-full-control"
        },
        "Bool": {
          "aws:SecureTransport": "true"
        }
      }
    },
    {
      "Sid": "AllowCloudFrontServicePrincipal",
      "Effect": "Allow",
      "Principal": {
        "Service": "cloudfront.amazonaws.com"
      },
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::eicheane-test-1026-2-ivs-recordings/*",
      "Condition": {
        "StringEquals": {
          "AWS:SourceArn": "arn:aws:cloudfront::844311324168:distribution/
E1NG4YMW5MN25A"
        }
      }
    }
  ]
}
```

Resolución de problemas

- La composición no está escrita en el bucket de S3: asegúrese de que el bucket de S3 y los objetos StorageConfiguration estén creados y estén en la misma región. Asegúrese también de que IVS tenga acceso al bucket consultando su política de bucket; consulte [Política de buckets para la configuración de almacenamiento](#).
- No encuentro una composición cuando ejecuto ListCompositions. Las composiciones son recursos efímeros. Una vez que pasan a un estado final, se eliminan automáticamente después de unos minutos.
- Mi composición se detiene automáticamente: una composición se detiene automáticamente si no hay ningún publicador en el escenario durante más de 60 segundos.

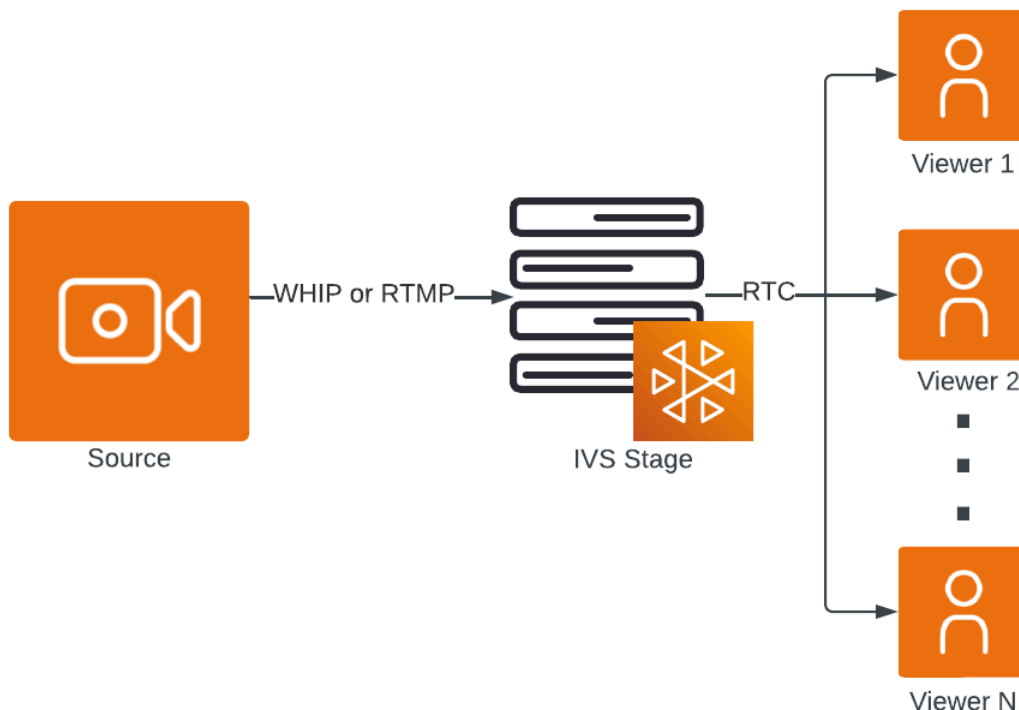
Problema conocido

La lista de reproducción multimedia escrita mediante grabación compuesta tiene la etiqueta #EXT-X-PLAYLIST-TYPE: EVENT mientras la composición está en curso. Una vez finalizada la composición, la etiqueta se actualiza a #EXT-X-PLAYLIST-TYPE: VOD. Para una experiencia de reproducción fluida, le recomendamos que utilice esta lista de reproducción solo después de que la composición se haya finalizado correctamente.

Ingesta de transmisiones de IVS | Transmisión en tiempo real

Como alternativa al uso del SDK de transmisión de IVS, puede publicar un video en una fase de IVS desde un origen WHIP o RTMP. Este enfoque ofrece flexibilidad para los flujos de trabajo en los que no es factible o preferible utilizar el SDK, como cuando se publica video desde OBS Studio o un codificador de hardware. Siempre que sea posible, recomendamos utilizar el SDK de transmisión de IVS, ya que no podemos garantizar el rendimiento ni la compatibilidad de las soluciones de terceros con IVS.

En este diagrama se ilustra cómo funciona la publicación con WHIP y RTMP:



Protocolos admitidos

Transmisión en tiempo real de IVS es compatible con varios protocolos de ingestión:

- RTMP (protocolo de mensajes en tiempo real): un estándar de la industria para la transmisión de video a través de una red.
- RTMPS: la versión segura de RTMP que funciona mediante TLS.

- WHIP (protocolo de ingesta WebRTC-HTTP): un borrador de IETF desarrollado para estandarizar la ingesta de WebRTC.

RTMP generalmente tiene una latencia más alta que WHIP, lo que lo hace ideal para transmisiones en directo de uno a varios. Para obtener una guía detallada sobre el uso de estos protocolos, consulte nuestra documentación sobre [RTMP](#) y [WHIP](#).

Especificaciones de contenido multimedia compatible

- Formato de entrada de audio
 - Códec: AAC-LC para RTMP y Opus para WHIP
 - Canales: 2 (estéreo) o 1 (mono)
 - Frecuencia de muestreo: 44,1 kHz o 48 kHz
 - Velocidad de bits máxima: 160 Kbps
- Formato de entrada de video
 - Códec: H.264
 - Perfil H.264: línea de base
 - Intervalo de IDR: 1 o 2 segundos
 - Velocidad de fotogramas: de 10 a 60 FPS
 - Fotogramas B: 0

Nota: El SDK de transmisión de IVS tiene los fotogramas B habilitados de forma predeterminada cuando se utiliza RTMP. Por lo tanto, los desarrolladores deben deshabilitar los marcos B: en iOS, utilice el método `usesBframes`; en Android, `setUseBframes`. Si los desarrolladores no deshabilitan los fotogramas B, sus transmisiones se desconectarán.

- Resolución máxima: 720p Mínimo: 160p
- Velocidad de bits máxima: 8,5 Mbps
- Configuración del codificador: recomendamos usar un codificador H.264 con los ajustes `veryfast` y `zerolatency`. Además, la opción `sliced_threads` de x264 está incluida en los ajustes preestablecidos de `zerolatency` y le recomendamos que la deshabilite. Por ejemplo, cuando utilice FFmpeg, su comando debería incluir: `-preset:v veryfast -tune zerolatency -x264-params sliced-threads=0`

Publicación RTMP de IVS | Transmisión en tiempo real

En este documento se describe el proceso de publicación en una fase de IVS mediante RTMP. Para obtener información adicional sobre las distintas opciones de ingesta, consulte la documentación

[Ingesta de transmisiones](#)

Creación de una fase

Utilice el siguiente comando para crear una fase:

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

Consulte [CreateStage](#) para obtener más información, incluida la respuesta.

Importante: En la respuesta, tenga en cuenta el campo `endpoints`, en el que se indican los puntos de conexión RTMP y RTMPS. Son necesarios para configurar el codificador RTMP.

Creación de una configuración de ingesta

Para publicar en una fase mediante RTMP, primero debe crear una configuración de ingesta y asociarla a la fase. Al publicar en la fase (con la clave de transmisión de la configuración de ingesta y el punto de conexión RTMP de la fase), el contenido multimedia se publicará en la fase como participante. Tiene la opción de especificar un valor de `userId` y valores de `attributes` personalizados, que se asociarán al [participante](#) que se conecte a la fase.

```
aws ivs-realtime create-ingest-configuration --name 'test' --stage-arn arn:aws:ivs:us-east-1:123456789012:stage/8faHz15Qp0ik --user-id = '123'
```

Consulte [CreateIngestConfiguration](#) para obtener más información, incluida la respuesta.

Al crear una configuración de ingesta, puede asociarla con el ARN de una fase específica por adelantado. Sin esta asociación, la clave de transmisión no se puede utilizar. Además, las configuraciones de ingesta (incluido el campo `stageArn`) se pueden actualizar mediante el punto de conexión [UpdateIngestConfiguration](#), lo que permite reutilizar la misma configuración para diferentes fases.

Nota: El campo `insecureIngest` de la configuración de ingesta tiene el valor predeterminado `false`, lo que requiere el uso de RTMPS. Se rechazarán las conexiones RTMP. Si debe usar RTMP,

establezca el valor de `insecureIngest` en `true`. Recomendamos usar RTMPS, a menos que tenga casos de uso específicos y verificados que requieran RTMP.

Publicación con un codificador RTMP

En este ejemplo se muestra cómo utilizar OBS Studio; sin embargo, puede utilizar cualquier codificador RTMP que cumpla con las [especificaciones de contenido multimedia](#) de IVS.

1. Descargue e instale el software: <https://obsproject.com/download>.
2. Haga clic en Settings (Configuración). En la sección Transmisión del panel Configuración, seleccione Personalizado en el menú desplegable Servicio.
3. En Servidor, ingrese el punto de conexión RTMP o RTMPS de la fase.
4. En Clave de transmisión, ingrese el valor de `streamKey` de la configuración de ingesta.
5. Configure los ajustes de video como lo haría normalmente, con algunas restricciones:
6.
 - a. Transmisión en tiempo real de IVS admite entradas de hasta 720p a 8,5 Mbps. Si supera cualquiera de estos límites, se desconectará su transmisión.
 - b. Le recomendamos configurar Intervalo de fotogramas clave en el panel Salida en 1 o 2 segundos. Un intervalo de fotogramas clave bajo permite a los espectadores iniciar la reproducción del video con mayor rapidez. También recomendamos configurar Ajuste preestablecido de uso de la CPU en muy rápido y Ajuste en latencia cero para habilitar la latencia más baja.
 - c. Como OBS no admite la transmisión simultánea, recomendamos mantener la velocidad de bits por debajo de 2,5 Mbps. Esto permite que los espectadores con conexiones de menor ancho de banda puedan ver los videos.
 - d. Deshabilite los fotogramas B, ya que las transmisiones con fotogramas B se desconectarán automáticamente. Realice una de las siguientes acciones siguientes:
 - En las opciones de x264, ingrese `bframes=0 sliced-threads=0`.
 - Establezca los fotogramas B en 0 si es una opción (por ejemplo, para NVENC).
7. Seleccione Iniciar transmisión.

Importante: Si la velocidad de bits máxima del codificador está establecida en 8,5 Mbps, el publicador desaparece ocasionalmente de la sesión. Esto se debe a que la configuración de velocidad de bits máxima es solo un objetivo y, en ocasiones, los codificadores lo sobrepasan. Para evitarlo, reduzca la velocidad de bits máxima del codificador, por ejemplo, a 6 Mbps.

Publicación WHIP de IVS | Transmisión en tiempo real

En este documento se explica cómo utilizar codificadores compatibles con WHIP, como OBS, para publicar transmisiones en tiempo real de IVS. [WHIP](#) (protocolo de ingesta WebRTC-HTTP) es un borrador de IETF desarrollado para estandarizar la ingesta de WebRTC.

WHIP habilita la compatibilidad con software como OBS y ofrece una alternativa (al SDK de transmisión de IVS) para la autoedición. Es posible que los streamers más sofisticados estén familiarizados con OBS y lo prefieran por sus características de producción avanzadas, como las transiciones de escenas, la mezcla de audio y la superposición de gráficos. Ofrece a los desarrolladores una opción versátil: usar el SDK de transmisión para web de IVS para publicar directamente en el navegador o permitir que los streamers usen OBS en su escritorio para obtener herramientas más potentes.

WHIP también resulta útil en situaciones en las que utilizar el SDK de transmisión de IVS no es viable o no es la opción preferida. Por ejemplo, en configuraciones que incluyen codificadores de hardware, el SDK de transmisión de IVS podría no ser una opción. Sin embargo, si el codificador es compatible con WHIP, puede seguir publicando directamente desde el codificador en IVS.

Guía para OBS

OBS es compatible con WHIP a partir de la versión 30. Para empezar, descargue la versión 30 o posterior de OBS: <https://obsproject.com/>.

Para publicar en una fase de IVS con OBS mediante WHIP, siga estos pasos:

1. [Genere](#) un token de participante con capacidad de publicación. En términos de WHIP, un token de participante es un token de portador. De forma predeterminada, los tokens de participantes vencen en 12 horas, pero puede ampliar su duración hasta 14 días.
2. Haga clic en Settings (Configuración). En la sección Transmisión del panel Configuración, seleccione WHIP en el menú desplegable Servicio.
3. En Servidor, ingrese <https://global.whip.live-video.net>.
4. En Token de portador, ingrese el token de participante que generó en el paso 2.
5. Configure los ajustes de video como lo haría normalmente, con algunas restricciones:
 - a. Transmisión en tiempo real de IVS admite entradas de hasta 720p a 8,5 Mbps. Si supera cualquiera de estos límites, se desconectará su transmisión.
 - b. Le recomendamos configurar Intervalo de fotogramas clave en el panel Salida en 1 o 2 segundos. Un intervalo de fotogramas clave bajo permite a los espectadores iniciar la

reproducción del video con mayor rapidez. También recomendamos configurar Ajuste preestablecido de uso de la CPU en muy rápido y Ajuste en latencia cero para habilitar la latencia más baja.

- c. Como OBS no admite la transmisión simultánea, recomendamos mantener la velocidad de bits por debajo de 2,5 Mbps. Esto permite que los espectadores con conexiones de menor ancho de banda puedan ver los videos.

6. Pulse Iniciar transmisión.

Nota: Somos conscientes de los problemas de calidad (como la congelación intermitente del video) que pueden producirse con WHIP en OBS. Suelen surgir cuando la red del difusor es inestable. Recomendamos probar WHIP en OBS antes de usarlo para transmisiones en vivo de producción. Reducir la velocidad de bits de la transmisión también puede ayudar a reducir la aparición de estos problemas.

Service Quotas de IVS | Transmisión en tiempo real

A continuación, se indican las cuotas de servicio y los límites para los puntos de conexión en tiempo real, los recursos y otras operaciones de Amazon Interactive Video Service (IVS). Las cuotas de servicio (que también se denominan límites) establecen el número máximo de recursos u operaciones de servicio que puede haber en una cuenta de AWS. Es decir, estos límites corren por cuenta de AWS, a menos que se indique lo contrario en la tabla. Consulte también [Service Quotas de AWS](#).

Para conectarse mediante programación a un servicio de AWS, utilice un punto de conexión. Consulte también [Puntos de conexión del servicio de AWS](#).

Todas las cuotas se aplican por región.

Aumentos en la cuota de servicio

Para cuotas ajustables, puede solicitar un aumento de la tasa a través de la [consola de AWS](#). También, utilice la consola para consultar información sobre cuotas de servicio.

Las cuotas de tarifa de llamadas de API no son ajustables.

Cuotas de tarifa de llamadas a la API

Tipo de punto de conexión	Punto de conexión	Predeterminado
Composición	GetComposition	5 TPS
Composición	ListCompositions	5 TPS
Composición	StartComposition	5 TPS
Composición	StopComposition	5 TPS
IngestConfiguration	CreateIngestConfiguration	5 TPS
IngestConfiguration	DeleteIngestConfiguration	5 TPS
IngestConfiguration	GetIngestConfiguration	5 TPS

Tipo de punto de conexión	Punto de conexión	Predeterminado
IngestConfiguration	ListIngestConfigurations	5 TPS
IngestConfiguration	UpdateIngestConfiguration	5 TPS
MediaEncoder	CreateEncoderConfiguration	5 TPS
MediaEncoder	DeleteEncoderConfiguration	5 TPS
MediaEncoder	GetEncoderConfiguration	5 TPS
MediaEncoder	ListEncoderConfigurations	5 TPS
PublicKey	DeletePublicKey	3 TPS
PublicKey	GetPublicKey	3 TPS
PublicKey	ImportPublicKey	3 TPS
PublicKey	ListPublicKeys	3 TPS
Escenario	CreateParticipantToken	50 TPS
Escenario	CreateStage	5 TPS
Escenario	DeleteStage	5 TPS
Escenario	DisconnectParticipant	5 TPS
Escenario	GetParticipant	5 TPS
Escenario	GetStage	5 TPS
Escenario	GetStageSession	5 TPS
Escenario	ListStages	5 TPS
Escenario	UpdateStage	5 TPS
Escenario	ListParticipants	5 TPS

Tipo de punto de conexión	Punto de conexión	Predeterminado
Escenario	ListParticipantEvents	5 TPS
Escenario	ListStageSessions	5 TPS
StorageConfiguration	CreateStorageConfiguration	5 TPS
StorageConfiguration	DeleteStorageConfiguration	5 TPS
StorageConfiguration	GetStorageConfiguration	5 TPS
StorageConfiguration	ListStorageConfigurations	5 TPS
Etiquetas	ListTagsForResource	10 TPS
Etiquetas	TagResource	10 TPS
Etiquetas	UntagResource	10 TPS

Otras cuotas

Recurso o característica	Predeterminado	Ajustable	Descripción
EncoderConfigurations	20	Sí	Número máximo de objetos EncoderConfiguration por cuenta.
Destinos de composición	2	No	Número máximo de objetos Destination en un objeto Composition.
Composición: duración máxima	24	No	Cantidad máxima de tiempo que puede existir una composición, en horas.

Recurso o característica	Predeterminado	Ajustable	Descripción
Composiciones	5	Sí	Cantidad máxima de objetos Composition simultáneos por cuenta.
IngestConfigurations	100	Sí	Número máximo de objetos IngestConfiguration por cuenta.
Velocidad de bits de publicación del participante	8,5 Mbps	No	Máximo de bits por segundo que se puede transmitir a una fase.
Duración de la publicación o suscripción de los participantes	24	No	Periodo máximo de tiempo durante el cual un participante puede publicar o permanecer suscrito a un escenario, en horas.
El participante publica la resolución	720p	No	Resolución máxima del video publicado por los participantes.
Tasa de bits de descarga de participantes	8,5 Mbps	No	Velocidad máxima de bits de descarga agregada en todas las suscripciones de un participante.
PublicKeys	3	No	Número máximo de claves públicas por región de AWS.
Participantes de la fase (publicadores)	12	No	Número máximo de participantes que pueden publicar en una fase a la vez.

Recurso o característica	Predeterminado	Ajustable	Descripción
Participantes de la fase (suscriptores)	10 000	Sí (hasta 25 000)	Número máximo de participantes que pueden suscribirse a una fase a la vez.
Escenarios	1 000	Sí	Número máximo de escenarios por región de AWS.
StorageConfigurations	5	Sí	Número máximo de objetos StorageConfiguration por cuenta.

Optimizaciones de transmisión en tiempo real de IVS

Para garantizar que sus usuarios tengan la mejor experiencia al transmitir y ver videos mediante la transmisión en tiempo real de IVS, hay varias formas de mejorar u optimizar partes de la experiencia con las características que ofrecemos actualmente.

Introducción

Al optimizar la calidad de la experiencia de un usuario, es importante tener en cuenta la experiencia deseada, que puede cambiar según el contenido que esté viendo y las condiciones de la red.

A lo largo de esta guía, nos centramos en los usuarios que son publicadores de transmisiones o suscriptores de transmisiones, y tenemos en cuenta las acciones y experiencias deseadas de esos usuarios.

Transmisión adaptativa: codificación en capas con transmisión simultánea

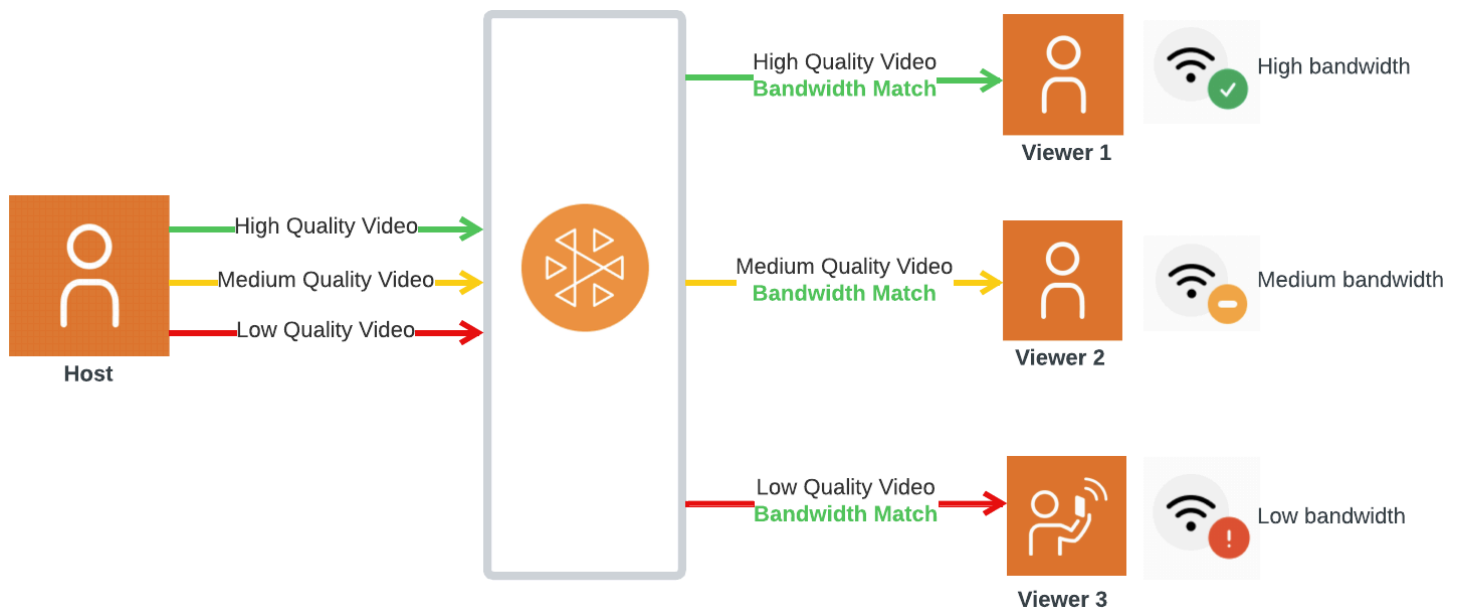
Esta característica solo se admite en las siguientes versiones de cliente:

- iOS y Android (versión 1.18.0 o superior)
- Web (versión 1.12.0 o superior)

Cuando se usan los [SDK de transmisión en tiempo real](#) de IVS, los publicadores pueden codificar varias capas de video y los suscriptores se adaptan automáticamente a la calidad que mejor se adapte a su red o la cambian. A esto lo llamamos codificación en capas con transmisión simultánea.

La codificación en capas con transmisión simultánea es compatible con Android y iOS y con los navegadores de escritorio Chrome y Edge (para Windows y macOS). No admitimos la codificación en capas en otros navegadores.

En el siguiente diagrama, el host envía tres calidades de video (alta, media y baja). IVS reenvía video de la más alta calidad a cada espectador en función del ancho de banda disponible. Esto proporciona una experiencia óptima para cada espectador. Si la conexión de red del espectador cambia de buena a mala, IVS comienza automáticamente a enviarle video de menor calidad, de modo que puede seguir viendo la transmisión sin interrupciones (con la mejor calidad posible).



Capas, calidades y velocidades de fotogramas predeterminadas

Las calidades y capas predeterminadas que se proporcionan a los usuarios de dispositivos móviles y de la web son las siguientes:

Móvil (Android, iOS)	Web (Chrome)
<p>Capa alta (o personalizada):</p> <ul style="list-style-type: none"> • Velocidad de bits máxima: 900 000 bps • Velocidad de fotogramas: 15 fps 	<p>Capa alta (o personalizada):</p> <ul style="list-style-type: none"> • Velocidad de bits máxima: 1 700 000 bps • Velocidad de fotogramas: 30 fps
<p>Capa intermedia: ninguna (no es necesaria, porque la diferencia entre las velocidades de bits de capa alta y baja en los dispositivos móviles es estrecha)</p>	<p>Capa intermedia:</p> <ul style="list-style-type: none"> • Velocidad de bits máxima: 700 000 bps • Velocidad de fotogramas: 20 fps
<p>Capa baja:</p> <ul style="list-style-type: none"> • Velocidad de bits máxima: 100 000 bps • Velocidad de fotogramas: 15 fps 	<p>Capa baja:</p> <ul style="list-style-type: none"> • Velocidad de bits máxima: 200 000 bps • Velocidad de fotogramas: 15 fps

Resolución de las capas

Las resoluciones de las capas media e inferior se reducen verticalmente de forma automática con respecto a la de la capa superior para mantener la misma relación de aspecto.

Se excluyen las capas media e inferior si sus resoluciones se acercan mucho a la de la capa superior. Por ejemplo, si la resolución configurada es de 320 x 180, el SDK no enviará también capas de menor resolución.

En la siguiente tabla se muestran las resoluciones de las capas generadas para diferentes resoluciones configuradas. Los valores proporcionados son para una orientación horizontal, pero se pueden aplicar en sentido inverso para el contenido que se transmita en vertical.

Resolución de entrada	Resoluciones de las capa de salida: móvil	Resoluciones de las capas de salida: web
720p (1280 x 720)	Alta (1280 x 720) Baja (320 x 180)	Alta (1280 x 720) Media (640 x 360) Baja (320 x 180)
540p (960 x 540)	Alta (960 x 540) Baja (320 x 180)	Alta (960 x 540) Baja (320 x 180)
360p (640 x 360)	Alta (640 x 360) Baja (360 x 180)	Alta (640 x 360) Baja (360 x 180)
270p (480 x 270)	Alta (480 x 270)	Alta (480 x 270)
180p (320 x 180)	Alta (320 x 180)	Alta (320 x 180)

Puede calcular las resoluciones de entrada personalizadas que no figuren en la lista anterior [con la siguiente herramienta](#).

Configuración de la codificación en capas con transmisión simultánea

Para utilizar la codificación por capas con transmisión simultánea, debe [haber habilitado la característica](#) en el cliente. Si la habilita, el publicador aumentará el uso del ancho de banda de carga, por lo que los espectadores experimentarán menos problemas de congelación de video.

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
```

Configuraciones de transmisión

En esta sección, se exploran otras configuraciones que puede aplicar a las transmisiones de video y audio.

Cambiar la velocidad de bits de la transmisión

Para cambiar la velocidad de bits de la transmisión, use los siguientes ejemplos de configuración.

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
```

Cambiar la velocidad de fotogramas de la transmisión de video

Para cambiar la velocidad de fotogramas de la transmisión de video, use los siguientes ejemplos de configuración.

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
```

Optimización de la tasa de bits de audio y el soporte estéreo

Para cambiar la velocidad de bits y la configuración estéreo de la transmisión de audio, use los siguientes ejemplos de configuración.

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
```

Cambio en el valor de MinDelay del búfer de fluctuación del suscriptor

Para cambiar el retardo mínimo del búfer de fluctuación de un participante que se está suscribiendo, se puede utilizar un valor de `subscribeConfiguration` personalizado. El búfer de fluctuación determina cuántos paquetes se almacenan antes de que comience la reproducción. El retardo mínimo representa el objetivo de la cantidad mínima de datos que se deben almacenar. Cambiar el retardo mínimo puede hacer que la reproducción sea más resiliente cuando se produzcan problemas de pérdida de paquetes o de conexión.

La desventaja de aumentar el tamaño del búfer de fluctuación es que también aumentará el retardo antes de que comience la reproducción. El aumento del retardo mínimo proporciona una mayor resiliencia, a costa de repercutir en el tiempo que se tarda en iniciar el video. Tenga en cuenta que aumentar el retardo mínimo durante la reproducción tiene un efecto similar: la reproducción se detendrá brevemente para permitir que se llene el búfer de fluctuación.

Si se necesita una mayor resiliencia, se recomienda empezar con un ajuste preestablecido de retardo mínimo de `MEDIUM` y definir la configuración de suscripción antes de que comience la reproducción.

Tenga en cuenta que el retardo mínimo se aplica solo si un participante es de suscripción única. Si un participante publica, no se aplica el retardo mínimo. Esto se hace para garantizar que varios publicadores puedan comunicarse entre sí sin retardos adicionales.

En los ejemplos siguientes se utiliza un ajuste preestablecido de retardo mínimo de `MEDIUM`. Consulte la documentación de referencia del SDK para ver todos los posibles valores.

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
}

```

Optimizaciones sugeridas

Escenario	Recomendaciones
Transmite con texto o contenido que se mueve lentamente, como presentaciones o diapositivas	Utilice la codificación por capas con la transmisión simultánea o configure las transmisiones con una velocidad de fotogramas más baja .
Transmisiones con acción o mucho movimiento	Utilice la codificación por capas con la transmisión simultánea .

Escenario	Recomendaciones
Transmisiones con conversación o poco movimiento	Utilice la codificación por capas con la transmisión simultánea o elija solo audio (consulte la sección “Suscripción a los participantes” en las guías del SDK de transmisiones en tiempo real: Web, Android e iOS).
Usuarios que transmiten con datos limitados	Utilice la codificación por capas con transmisión simultánea o, si desea reducir el uso de datos para todos, configure una velocidad de fotogramas más baja y baje la velocidad de bits manualmente .

Recursos y asistencia de IVS | Transmisión en tiempo real

Este documento ofrece una lista de recursos que sirven de ayuda para utilizar la transmisión en tiempo real de Amazon IVS.

Recursos

<https://ivs.rocks/> es un sitio dedicado a explorar el contenido publicado (demostraciones, ejemplos de código, publicaciones de blog), calcular el costo y experimentar Amazon IVS a través de demostraciones en vivo.

Demostraciones



La demostración de streaming en tiempo real de IVS para iOS y Android muestra a los desarrolladores cómo utilizar Amazon IVS para crear una atractiva aplicación de contenido en tiempo real generada por usuarios de redes sociales. Esta aplicación incluye una fuente desplazable de

transmisiones en tiempo real generadas por los usuarios. Los usuarios pueden crear transmisiones de video y salas solo de audio. Los invitados que hagan streaming de video pueden unirse en modo invitado o en modo versus (VS). Las instrucciones sobre cómo implementar el backend requerido y crear la aplicación están disponibles en los siguientes repositorios de 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: <https://github.com/aws-samples/amazon-ivs-real-time-serverless-demo/>

Soporte

El [Centro de AWS](#) ofrece una serie de planes que proporcionan acceso a herramientas y conocimientos para dar soporte a sus soluciones de AWS. Todos los planes de soporte proporcionan acceso ininterrumpido a atención al cliente. Para obtener asistencia técnica y recursos adicionales para planificar, implementar y mejorar su entorno de AWS, seleccione un plan de asistencia técnica acorde con su caso de uso de AWS.

[AWS Premium Support](#) es un canal de asistencia individualizado y de respuesta rápida que le ayudará a crear y ejecutar aplicaciones en AWS.

[AWS re:Post](#) es un sitio de preguntas y respuestas para la comunidad de desarrolladores donde se tratan aspectos técnicos relacionados con Amazon IVS.

En [Contáctese con AWS](#) tiene enlaces para hacernos llegar sus preguntas no técnicas sobre facturación o su cuenta. Para preguntas técnicas, utilice los foros de debate o los enlaces de soporte previamente proporcionados.

Glosario de IVS

Consulte también el [glosario de AWS](#). En la siguiente tabla, LL son las siglas de transmisión de baja latencia de IVS; RT, transmisión en tiempo real de IVS.

Plazo	Descripción	LL	RT	Chat
AAC	Advanced Audio Coding. AAC es un estándar de codificación de audio para la compresión de audio digital con pérdida. Diseñado para ser el sucesor del formato MP3, AAC generalmente logra una calidad de sonido superior a este con la misma velocidad de bits. El formato AAC fue estandarizado por la ISO y la IEC como parte de las especificaciones MPEG-2 y MPEG-4.	✓	✓	
Transmisión con velocidad de bits adaptable	La transmisión con velocidad de bits adaptable (ABR) permite al reproductor de IVS cambiar a una velocidad de bits más baja cuando la calidad de la conexión se vea afectada y volver a una velocidad más alta cuando la calidad mejore.	✓		
Transmisión adaptativa	Consulte Codificación por capas con transmisión simultánea .		✓	
Usuario administrativo	Un usuario de AWS con acceso administrativo a los recursos y servicios disponibles en una cuenta de AWS. Consulte Terminología en la Guía del usuario de configuración de AWS.	✓	✓	✓
ARN	Nombre de recurso de Amazon , un identificador exclusivo de cualquier recurso de AWS. Los formatos específicos de ARN dependen del tipo de recurso. Para conocer los formatos de ARN utilizados por los recursos de IVS, consulte la referencia de autorización de servicio.	✓	✓	✓

Plazo	Descripción	LL	RT	Chat
Relación de aspecto	Describe la relación entre el ancho y la altura del marco. Por ejemplo, 16:9 es la relación de aspecto que corresponde a la resolución Full HD o 1080p.	✓	✓	
Modo de audio	Una configuración de audio preestablecida o personalizada que está optimizada para diferentes tipos de usuarios de dispositivos móviles y los equipos que usan. Consulte SDK de transmisión de IVS: modos de audio móvil (streaming en tiempo real) .		✓	
AVC, H.264, MPEG-4 parte 10	La codificación de video avanzada, también denominada H.264 o MPEG-4 parte 10, es un estándar para la compresión de video digital con pérdida.	✓	✓	
Reemplazo de fondo	Un tipo de filtro de cámara que permite a los creadores de transmisiones en directo cambiar sus fondos. Consulte Reemplazo de fondo en SDK de transmisión de IVS: filtros de cámara de terceros (streaming en tiempo real).		✓	
Velocidad de bits	Métrica de transmisión que indica el número de bits que se transmiten o reciben por segundo.	✓	✓	
Difusión, emisora	Otros términos para transmisión o streamer .	✓		

Plazo	Descripción	LL	RT	Chat
Almacenamiento en búfer	Una condición que se produce cuando el dispositivo de reproducción no puede descargar el contenido antes de que este se reproduzca. El almacenamiento en búfer puede manifestarse de varias formas: el contenido puede detenerse y comenzar de forma aleatoria (lo que también se conoce como inestabilidad), el contenido puede detenerse durante periodos prolongados (también conocido como congelación) o el reproductor de IVS se puede pausar.	✓	✓	
Listas de reproducción por rango de bytes	<p>Una lista de reproducción más detallada que la lista de reproducción HLS estándar. La lista de reproducción HLS estándar se compone de archivos multimedia de 10 segundos. Con una lista de reproducción por rango de bytes, la duración del segmento es la misma que el intervalo de fotogramas clave configurado para la transmisión.</p> <p>La lista de reproducción por rango de bytes solo está disponible para las emisiones que se grabaron automáticamente en un bucket de S3. Se crea además de la lista de reproducción HLS. Consulte Listas de reproducción por rango de bytes en Grabación automática en Amazon S3 (streaming de baja latencia).</p>	✓		

Plazo	Descripción	LL	RT	Chat
CBR	Velocidad de bits constante (CBR): un método de control de velocidad para codificadores que mantiene una velocidad de bits constante durante toda la reproducción de un video, sin importar lo que ocurra durante la transmisión. Las pausas en la acción se pueden rellenar para alcanzar la velocidad de bits deseada. De igual manera, los picos se pueden cuantificar al ajustar la calidad de la codificación para que coincida con la velocidad de bits objetivo. Recomendamos encarecidamente utilizar CBR en lugar de VBR .	✓	✓	
CDN	Red de entrega de contenidos o red de distribución de contenidos: una solución distribuida de manera geográfica que optimiza la distribución de contenidos, como la transmisión de video, y la acerca al lugar donde se encuentran los usuarios.	✓		
Canal	Un recurso de IVS que almacena la configuración para la transmisión, lo que incluye un servidor de ingesta , una clave de transmisión , una URL de reproducción y opciones de grabación. Los streamers utilizan la clave de transmisión asociada a un canal para iniciar una difusión. Todas las métricas y eventos están asociados a un recurso de canal.	✓		
Tipo de canal	Determina la resolución y la velocidad de fotogramas permitidas para el canal . Consulte Tipos de canales en la Referencia de la API de transmisión de baja latencia de IVS.	✓		
Registro del chat	Una opción avanzada que se puede habilitar mediante la vinculación de una configuración de registro con una sala de chat .			✓

Plazo	Descripción	LL	RT	Chat
Sala de chat	Recurso de IVS que almacena la configuración de una sesión de chat e incluye funciones opcionales como el controlador de revisión de mensajes y el registro de conversaciones . Consulte Step 2: Create a Chat Room en Getting Started with IVS Chat.			✓
Composición del cliente	Usa un dispositivo host para mezclar las transmisiones de audio y video de los participantes del escenario y, a continuación, las envía como una transmisión compuesta a un canal de IVS. Esto permite un mayor control sobre el aspecto de la composición , pero a costa de una mayor utilización de los recursos del cliente y un mayor riesgo de que un problema con el escenario o el host afecte a los espectadores. Consulte también Composición del servidor .	✓	✓	
CloudFront	Un servicio de CDN que proporciona Amazon.	✓		
CloudTrail	Un servicio de AWS para recopilar, supervisar, analizar y retener los eventos y la actividad de las cuentas de AWS, así como de orígenes externos. Consulte Registro de llamadas a la API de IVS con AWS CloudTrail .	✓	✓	✓
CloudWatch	Un servicio de AWS para supervisar las aplicaciones, responder a los cambios de rendimiento, optimizar el uso de los recursos y proporcionar información sobre el estado de las operaciones. Puede usar CloudWatch para supervisar las métricas de IVS; consulte Supervisión del streaming en tiempo real de IVS y Monitoreo de transmisión de baja latencia de Amazon IVS .	✓	✓	✓

Plazo	Descripción	LL	RT	Chat
Composición	Proceso que consiste en combinar transmisiones de audio y video de varios orígenes en una sola transmisión.	✓	✓	
Canalización de composición	Secuencia de pasos de procesamiento necesaria para combinar varias transmisiones y codificar la resultante.	✓	✓	
Compresión	Codificación de la información con menos bits que la representación original. Cualquier compresión en particular puede ser con o sin pérdida. La compresión sin pérdida reduce los bits al identificar y eliminar la redundancia estadística. No se pierde información en la compresión sin pérdida. La compresión con pérdida reduce los bits al eliminar información innecesaria o menos importante.	✓	✓	
Plano de control	Almacena información sobre los recursos de IVS, como canales , escenarios o salas de chat . Además, proporciona interfaces para crear y administrar estos recursos. Es regional (se basa en las regiones de AWS).	✓	✓	✓
CORS	Uso compartido de recursos entre orígenes: una característica que permite a las aplicaciones web de los clientes cargadas en un dominio interactuar con los recursos (como buckets de S3) de un dominio diferente. El acceso se puede configurar en función de los encabezados, los métodos HTTP y los dominios de origen. Consulte Uso compartido o de recursos entre orígenes (CORS): Amazon Simple Storage Service en la Guía del usuario de Amazon Simple Storage Service.	✓		

Plazo	Descripción	LL	RT	Chat
Origen de imagen personalizado	Una interfaz proporcionada por el SDK de transmisión de IVS que permite que una aplicación proporcione su propia entrada de imagen en vez de limitarse a las cámaras predeterminadas.	✓	✓	
Plano de datos	La infraestructura que transmite los datos desde la ingesta hasta la salida. Funciona según la configuración administrada en el plano de control y no está restringida a una región de AWS.	✓	✓	✓
Codificador, codificación	Proceso de convertir contenido de video y audio a un formato digital que sea adecuado para la transmisión. La codificación puede estar basada en hardware o software.	✓	✓	
Evento	Notificación automática publicada por IVS en el servicio de supervisión de AmazonEventBridge. Un evento representa un cambio de estado de un recurso de transmisión, como un escenario o una canalización de composición . Consulte Uso de Amazon EventBridge con el streaming de baja latencia de IVS y Uso de Amazon EventBridge con el streaming en tiempo real de IVS .	✓	✓	✓
FFmpeg	Un proyecto de software libre y de código abierto que consiste en un conjunto de bibliotecas y programas para gestionar archivos y transmisiones de audio y video. FFmpeg proporciona una solución compatible con diversas plataformas para grabar, convertir y transmitir audio y video.	✓		

Plazo	Descripción	LL	RT	Chat
Transmisión fragmentada	Se crea cuando una transmisión se desconecta y, a continuación, se vuelve a conectar dentro del intervalo especificado en la configuración de grabación del canal . Las diversas transmisiones resultantes se consideran una sola, por lo que se combinan en una sola transmisión grabada. Consulte Fusionar transmisiones fragmentadas en Grabación automática en Amazon S3 (streaming de baja latencia).	✓		
Velocidad de fotogramas	Métrica de transmisión que indica el número de fotogramas que se transmiten o reciben por segundo.	✓	✓	
HLS	HTTP Live Streaming (HLS): un protocolo de comunicaciones de transmisión con velocidad de bits adaptable que se basa en HTTP y se utiliza para entregar transmisiones de IVS a los espectadores.	✓		
Lista de reproducción HLS	Lista de segmentos multimedia que componen una transmisión. Las listas de reproducción HLS estándar se componen de archivos multimedia de 10 segundos. HLS también admite listas de reproducción con un rango de bytes más granular .	✓		
Host	Participante de un evento en tiempo real que envía video o audio al escenario.		✓	
IAM	Identity and Access Management: un servicio de AWS que permite a los usuarios administrar de forma segura las identidades y el acceso a los servicios y recursos de AWS, incluido IVS.	✓	✓	✓

Plazo	Descripción	LL	RT	Chat
Incorporación	Proceso de IVS que consiste en recibir transmisiones de video de un host o difusor para su procesamiento o entrega a los espectadores u otros participantes.	✓	✓	
Servidor de incorporación	<p>Recibe transmisiones de video y las envía a un sistema de transcodificación, en el que las transmisiones se multiplexan o transcodifican en HLS para su entrega a los espectadores.</p> <p>Los servidores de ingesta son componentes específicos de IVS y reciben las transmisiones de los canales, así como un protocolo de ingesta (RTMP, RTMPS). Consulte la información sobre cómo crear un canal en Introducción al streaming de baja latencia de IVS.</p>		✓	
Video entrelazado	Transmite y muestra solo las líneas pares o impares de los fotogramas posteriores para duplicar la velocidad de fotogramas percibida sin consumir más ancho de banda. No recomendamos utilizar video entrelazado debido a problemas en su calidad.	✓	✓	
JSON	JavaScript Object Notation: formato de archivo estándar abierto que utiliza texto legible por humanos para transmitir objetos de datos que consisten en pares de valor de atributo y tipos de datos de matriz o cualquier otro valor serializable.	✓	✓	✓

Plazo	Descripción	LL	RT	Chat
Fotograma clave, fotograma delta, intervalo de fotogramas clave	El fotograma clave (también denominado intracodificado o fotograma i) es un fotograma completo de la imagen de un video. Los fotogramas posteriores, los fotogramas delta (también denominados fotogramas previstos o fotogramas p), solo contienen la información que ha cambiado. Los fotogramas clave aparecerán varias veces dentro de una transmisión , según el intervalo de fotogramas clave definido en el codificador.	✓	✓	
Lambda	Un servicio de AWS para ejecutar código (denominadas funciones de Lambda) sin aprovisionar ninguna infraestructura de servidor. Las funciones de Lambda se pueden ejecutar en respuesta a eventos y solicitudes de invocación o según una programación. Por ejemplo, Chat de IVS utiliza funciones de Lambda para permitir la revisión de mensajes en una sala de chat .	✓	✓	✓
Latencia, latencia integral	Un retraso en la transferencia de datos. IVS define los rangos de latencia de la siguiente manera: <ul style="list-style-type: none"> • Baja latencia: menos de 3 segundos • Latencia en tiempo real: menos de 300 ms <p>La latencia integral se refiere al retraso desde el momento en que una cámara captura una transmisión en vivo hasta el momento en que la transmisión aparece en la pantalla de un espectador.</p>	✓	✓	
Codificación por capas con la transmisión simultánea	Permite la codificación y publicación simultáneas de varias transmisiones de video con distintos niveles de calidad. Consulte Transmisión adaptativa: codificación en capas con transmisión simultánea y Optimizaciones de streaming en tiempo real.		✓	

Plazo	Descripción	LL	RT	Chat
Controlador de revisión de mensajes	Permite a los clientes de Chat de IVS revisar y filtrar automáticamente los mensajes de los usuarios antes de enviarlos a la sala de chat . Se habilita al asociar una función de Lambda con una sala de chat. Consulte Creating a Lambda Function en Chat Message Review Handler .			✓
Mezclador	Una característica de los SDK de transmisión móvil de IVS que recibe varios orígenes de audio y video y genera una única salida. Admite la administración de los elementos de video y audio que están en pantalla y que representan orígenes como cámaras, micrófonos, capturas de pantalla, así como audio y video generados por la aplicación. Luego, la salida se puede transmitir a IVS. Consulte Configuración de una sesión de transmisión de mezcla en SDK de transmisión de IVS: guía del mezclador (streaming de baja latencia).	✓		
Transmisión de varios hosts	Combina transmisiones de varios hosts en una sola transmisión. Esto se puede lograr mediante una composición del cliente o del servidor . La transmisión de varios hosts permite, por ejemplo, invitar a los espectadores a un escenario para una sesión de preguntas y respuestas, concursos entre hosts, videoconferencias y hosts conversando entre sí frente a un gran público.		✓	
Lista de reproducción multivariante	Un índice de todas las transmisiones variantes disponibles para una transmisión.	✓		
OAC	Origin Access Control: un mecanismo para restringir el acceso a un bucket de S3 , de modo que el contenido, como una transmisión grabada, solo se pueda ofrecer a través de la CDN de CloudFront .	✓		

Plazo	Descripción	LL	RT	Chat
OBS	Open Broadcaster Software: software gratuito y de código abierto para grabación y transmisión en directo de video. OBS ofrece una alternativa (al SDK de transmisión de IVS) para la autoedición. Es posible que los streamers más sofisticados estén familiarizados con OBS y lo prefieran por sus características de producción avanzadas, como las transiciones de escenas, la mezcla de audio y la superposición de gráficos.	✓	✓	
Participante	Un usuario en tiempo real conectado al escenario como host o espectador .		✓	
Token de participante	Autentica a un participante del evento en tiempo real cuando se une a un escenario . Un token de participante también controla si un participante puede enviar video al escenario.		✓	
Token de reproducción, par de claves de reproducción	<p>Un mecanismo de autorización que permite a los clientes restringir la reproducción de video en canales privados. Los tokens de reproducción se generan a partir de un par de claves de reproducción.</p> <p>Un par de claves de reproducción es el par de claves público-privadas utilizado para firmar y validar el token de autorización del espectador para la reproducción. Consulte Crear o importar una clave de reproducción de IVS en Configuración de canales privados de IVS y consulte los puntos de conexión del par de claves de reproducción en la referencia de la API de baja latencia de IVS.</p>	✓		

Plazo	Descripción	LL	RT	Chat
URL de reproducción	Identifica la dirección que utiliza el espectador para iniciar la reproducción de un canal específico. Esta dirección se puede utilizar globalmente. IVS selecciona automáticamente la mejor ubicación de la red global de distribución de contenido de IVS para entregar el video a cada espectador . Consulte la información sobre cómo crear un canal en Introducción al streaming de baja latencia de IVS .	✓		
Canal privado	Permite a los clientes restringir el acceso a sus transmisiones mediante un mecanismo de autorización que se basa en los tokens de reproducción . Consulte Flujo de trabajo para canales privados de IVS en Configuración de canales privados de IVS.	✓		
Video progresivo	Transmite y muestra todas las líneas de cada fotograma en secuencia. Recomendamos utilizar video progresivo durante todas las etapas de la transmisión.	✓	✓	
Cuotas	El número máximo de recursos u operaciones de servicio de IVS que puede haber en una cuenta de AWS. Es decir, estos límites corren por cuenta de AWS, a menos que se indique lo contrario. Todas las cuotas se aplican por región. Consulte Puntos de conexión y cuotas de Amazon Interactive Video Service en la Guía de referencia general de AWS.	✓	✓	✓

Plazo	Descripción	LL	RT	Chat
Regiones	<p>Otorga acceso a servicios de AWS que se ubican físicamente en un área geográfica determinada. Las regiones proporcionar tolerancia a errores, estabilidad y resistencia, y también pueden reducir la latencia. Con las regiones, puede crear recursos redundantes que sigan estando disponibles y no resulten afectados por una interrupción regional.</p> <p>La mayoría de las solicitudes de servicio de AWS están asociadas a una región geográfica en particular. Los recursos que crea en una región no existen en ninguna otra región salvo que utilice explícitamente una característica de replicación ofrecida por un servicio de AWS. Por ejemplo, Amazon S3 admiten la replicación entre regiones. Algunos servicios, como IAM, no tienen recursos entre regiones.</p>	✓	✓	✓
Resolución	Describe la cantidad de píxeles en un único fotograma de video; por ejemplo, Full HD o 1080p definen un fotograma con 1920 x 1080 píxeles.	✓	✓	
Usuario raíz	El propietario de una cuenta de AWS. Este usuario raíz tiene acceso a todos los recursos y los servicios de AWS en la cuenta de AWS.	✓	✓	✓
RTMP, RTMPS	Protocolo de mensajes en tiempo real: un estándar del sector para la transmisión de audio, video y datos a través de una red. RTMPS es la versión segura de RTMP, que se ejecuta a través de una conexión de seguridad de la capa de transporte (TLS/SSL).	✓	✓	

Plazo	Descripción	LL	RT	Chat
Bucket de S3	Un conjunto de objetos almacenados en Amazon S3. Muchas políticas, como las de acceso y replicación, se definen en el nivel de bucket y se aplican a todos los objetos del bucket. Por ejemplo, una transmisión de IVS se almacena como varios objetos en un bucket de S3.	✓		
SDK	Kit de desarrollo de software: un conjunto de bibliotecas para los desarrolladores que crean aplicaciones con IVS.	✓	✓	✓
Segmentación de selfies	Permite sustituir el fondo en una transmisión en directo mediante una solución específica para el cliente que acepta una imagen de la cámara como entrada y devuelve una máscara que proporciona una puntuación de confianza para cada píxel de la imagen, que indica si está en primer plano o en segundo. Consulte Reemplazo de fondo en SDK de transmisión de IVS: filtros de cámara de terceros (streaming en tiempo real).		✓	
Control de versiones semántico	Formato de versión con el formato VersiónPrincipal.VersiónSecundaria.Revisión. Las correcciones de errores que no afectan a la API aumentan la versión de revisión, las adiciones o cambios de la API compatibles con versiones anteriores aumentan la versión secundaria y los cambios de la API incompatibles con versiones anteriores aumentan la versión principal.	✓	✓	✓

Plazo	Descripción	LL	RT	Chat
Composición del servidor	<p>Utiliza un servidor de IVS para mezclar el audio y el video de todos los participantes del escenario y, a continuación, envía este video mezclado a un canal de IVS (por ejemplo, para llegar a un público más amplio) o a un bucket de S3. La composición del servidor reduce la carga de clientes, mejora la resiliencia de la transmisión y permite un uso más eficiente del ancho de banda.</p> <p>Consulte también Composición del cliente.</p>		✓	
Service Quotas	<p>Un servicio de AWS que lo ayuda a administrar las cuotas de muchos servicios de AWS desde una sola ubicación. Además de buscar los valores de cuota, también puede solicitar un aumento de cuota desde la consola de Service Quotas.</p>	✓	✓	✓
Rol vinculado al servicio	<p>Un tipo único de rol de IAM que está vinculado directamente a un servicio de AWS. IVS crea automáticamente los roles vinculados a servicios e incluyen todos los permisos que el servicio requiere para llamar a otros servicios de AWS en su nombre, por ejemplo, a un bucket de S3. Consulte Uso de roles vinculados a servicios para IVS en Seguridad de IVS.</p>	✓		
Escenario	<p>Un recurso de IVS que representa un espacio virtual en el que los participantes del evento en tiempo real pueden intercambiar videos también en tiempo real. Consulte Creación de un escenario en Introducción a streaming en tiempo real de IVS.</p>		✓	

Plazo	Descripción	LL	RT	Chat
Sesión de escenario	La sesión de un escenario comienza cuando el primer participante se une a un escenario y finaliza unos minutos después de que el último participante deje de publicar en él. Un escenario de larga duración puede tener varias sesiones a lo largo de su vida útil.		✓	
De streaming	Datos que representan contenido de video o audio que se envían de forma continua de un origen a un destino.	✓	✓	
Clave de transmisión	Identificador asignado por IVS cuando se crea un canal y se utiliza para autorizar la transmisión. Trate la clave de transmisión como un secreto, ya que permite a cualquiera que la tenga transmitir en el canal. Consulte Introducción al streaming de baja latencia de IVS .	✓		
Transmisión en inanición	<p>Un retraso o una interrupción en la entrega de la transmisión a IVS. Se produce cuando IVS no recibe la cantidad esperada de bits que el dispositivo de codificación anunció que enviaría en un periodo determinado. Si se produce una transmisión en inanición, se produce un evento de transmisión en inanición.</p> <p>Desde la perspectiva del espectador, la transmisión en inanición puede manifestarse como un video que se retrasa, se almacena en búfer o se congela. La transmisión en inanición puede ser breve (menos de 5 segundos) o larga (varios minutos), según la situación específica que la provocó. Consulte ¿Qué es la transmisión en inanición? en Preguntas frecuentes de solución de problemas.</p>	✓	✓	

Plazo	Descripción	LL	RT	Chat
Streamer	Persona o dispositivo que envía una transmisión de video o audio a IVS.	✓	✓	
Suscriptor	Participante de un evento en tiempo real que recibe video o audio del host. Consulte ¿Qué es el streaming en tiempo real de IVS? .		✓	
Etiqueta	Una etiqueta de metadatos que asigna a un recurso de AWS. Las etiquetas pueden ayudarlo a identificar y organizar sus recursos de AWS. En la página de inicio de la documentación de IVS , consulte la sección “Etiquetado” en cualquier documentación de la API de IVS (para la transmisión en tiempo real, la transmisión de baja latencia o el chat).	✓	✓	✓
Filtros de cámara de terceros	Componentes de software que se pueden integrar con el SDK de transmisión de IVS para permitir que una aplicación procese las imágenes antes de proporcionarlas este como origen de imágenes personalizado . Un filtro de cámara de terceros puede procesar las imágenes de la cámara, aplicar un efecto de filtro, etc.	✓	✓	
Miniatura	Imagen de tamaño reducido tomada de una transmisión. De forma predeterminada, las miniaturas se generan cada 60 segundos, pero se puede configurar un intervalo más corto. La resolución de las miniaturas depende del tipo de canal . Consulte Grabación de contenidos en Grabación automática en Amazon S3 (streaming de baja latencia).	✓		

Plazo	Descripción	LL	RT	Chat
Metadatos cronometrados	<p>Metadatos vinculados a marcas de tiempo específicas en una transmisión. Se pueden agregar mediante programación con la API de IVS y se asocian a fotogramas específicos. Esto garantiza que todos los espectadores reciben los metadatos en el mismo punto de la transmisión.</p> <p>Los metadatos cronometrados se pueden utilizar para activar acciones en el cliente, como actualizar las estadísticas del equipo durante un evento deportivo. Consulte Incorporación de metadatos en una transmisión de video.</p>	✓		
Transcodificación	<p>Convierte video y audio de un formato a otro. Una transmisión entrante puede ser transcodificada a un formato diferente con múltiples velocidades de bits y resoluciones, para admitir una variedad de dispositivos de reproducción y condiciones de red.</p>	✓	✓	
Transmuxing	<p>Un simple reempaquetado de una transmisión ingerida a Amazon IVS, sin volver a codificar la transmisión de video. “Transmux” es la abreviatura en inglés de multiplexación de transcodificación, un proceso que cambia el formato de un archivo de audio o video manteniendo algunas o todas las transmisiones originales. Transmuxing se convierte a un formato de contenedor diferente sin cambiar el contenido del archivo. Se distingue de la transcodificación.</p>	✓	✓	

Plazo	Descripción	LL	RT	Chat
Transmisiones variantes	<p>Conjunto de codificaciones de la misma transmisión en distintos niveles de calidad. Cada transmisión variante se codifica como una lista de reproducción HLS independiente. Un índice de las transmisiones variantes disponibles se denomina lista de reproducción multivariante.</p> <p>Una vez que el reproductor de IVS recibe una lista de reproducción multivariante de IVS, puede elegir entre las transmisiones variantes durante la reproducción y cambiar sin problemas entre ellas a según cambien las condiciones de la red.</p>	✓		
VBR	<p>Velocidad de bits variable (VBR): un método de control de velocidad para codificadores que utiliza una velocidad de bits dinámica que cambia durante la reproducción en función del nivel de detalle necesario. Recomendamos encarecidamente no utilizar VBR debido a problemas con la calidad de video; en su lugar, utilice CBR.</p>	✓	✓	

Plazo	Descripción	LL	RT	Chat
Vista	<p>Una sesión de visualización única que descarga o reproduce videos de forma activa. Las vistas son la base de la cuota de vistas simultáneas.</p> <p>Una vista se inicia cuando una sesión de visualización comienza la reproducción de video. Una vista finaliza cuando una sesión de visualización detiene la reproducción de video. La reproducción es el único indicador de la audiencia; no se tienen en cuenta las heurísticas de interacción, como los niveles de audio, el enfoque de la pestaña del navegador y la calidad del video. Al contar las vistas, IVS no tiene en cuenta la legitimidad de los espectadores individuales ni intenta deduplicar la audiencia localizada, como varios reproductores de video en un solo equipo. Consulte Otras cuotas en Service Quotas (streaming de baja latencia).</p>	✓		
Visor	Una persona que recibe una transmisión de IVS.	✓		
WebRTC	<p>Web Real-Time Communication: un proyecto de código abierto que proporciona comunicación en tiempo real a los navegadores web y aplicaciones móviles. Permite que la comunicación de audio y video funcione dentro de las páginas web al permitir la comunicación directa entre pares, lo que elimina la necesidad de instalar complementos o descargar aplicaciones nativas.</p> <p>Las tecnologías detrás de WebRTC se implementan como un estándar web abierto y están disponibles como API de JavaScript normales en los principales navegadores o como bibliotecas para clientes nativos (por ejemplo, Android e iOS).</p>	✓	✓	

Plazo	Descripción	LL	RT	Chat
WHIP	<p>Protocolo de ingesta HTTP WebRTC, un protocolo basado en HTTP que permite la ingesta de contenido basada en WebRTC en servicios de transmisión o CDN. WHIP es un borrador del IETF desarrollado para estandarizar la ingesta de WebRTC.</p> <p>WHIP habilita la compatibilidad con software como OBS y ofrece una alternativa (al SDK de transmisión de IVS) para la autoedición. Es posible que los streamers más sofisticados estén familiarizados con OBS y lo prefieran por sus características de producción avanzadas, como las transiciones de escenas, la mezcla de audio y la superposición de gráficos.</p> <p>WHIP también resulta útil en situaciones en las que utilizar el SDK de transmisión de IVS no es viable o no es la opción preferida. Por ejemplo, en configuraciones que incluyen codificadores de hardware, el SDK de transmisión de IVS podría no ser una opción. Sin embargo, si el codificador es compatible con WHIP, puede seguir publicando directamente desde el codificador en IVS.</p> <p>Consulte Compatibilidad de IVS con WHIP (streaming en tiempo real).</p>		✓	
WSS	<p>WebSocket Secure: un protocolo para establecer WebSockets a través de una conexión TLS cifrada. Se utiliza para conectarse a los puntos de conexión de Chat de IVS. Consulte Step 4: Send and Receive Your First Message en Getting Started with IVS Chat.</p>			✓

Historial de documentos de IVS | Transmisión en tiempo real

En las siguientes tablas se detallan los cambios importantes que se han realizado en la documentación de Transmisión en tiempo real de Amazon IVS. Actualizamos la documentación con frecuencia, para nuevas versiones y para dar respuesta a los comentarios que se nos envían.

Cambios en la Guía del usuario de transmisión en tiempo real

Cambio	Descripción	Fecha
SDK de transmisión: versión 1.17.0 para web	Se han actualizado el número de versión y los enlaces de los artefactos en la página de inicio de la documentación de IVS y en la guía del SDK de transmisión de baja latencia: web . Consulte también Notas de la versión .	10 de octubre de 2024
SDK de transmisión: versión 1.23.0 para Android y versión 1.23.0 para iOS	Se han actualizado el número de versión y los enlaces de los artefactos en la página de inicio de la documentación de IVS y en las guías del SDK de transmisión en tiempo real: Android y iOS . Consulte también Notas de la versión . En el caso de Android, hemos agregado Uso del SDK con símbolos de depuración .	10 de octubre de 2024
Service Quotas	Hemos agregado una cuota para la “velocidad de bits de publicación del participante”.	25 de septiembre de 2024

[Supervisión de transmisión en tiempo real de IVS](#)

Se agregó la métrica PublishFramerate de CloudWatch.

13 de septiembre de 2024

[SDK de transmisión: versión 1.22.0 para Android y versión 1.22.0 para iOS](#)

Se han actualizado el número de versión y los enlaces de los artefactos en la [página de inicio de la documentación de IVS](#) y en las guías del SDK de transmisión en tiempo real: [Android](#) y [iOS](#). Consulte también [Notas de la versión](#).

11 de septiembre de 2024

También hemos actualizado la sección “Introducción > Instalación de la biblioteca” en la guía del SDK de transmisión para Android.

[SDK de transmisión: versión 1.16.0 para web](#)

Se han actualizado el número de versión y los enlaces de los artefactos en la [página de inicio de la documentación de IVS](#) y en la guía del SDK de transmisión de baja latencia: [web](#). Consulte también [Notas de la versión](#).

11 de septiembre de 2024

[Ingesta RTMP](#)

Hemos agregado la página [Ingesta de transmisiones de IVS](#). En esta hay dos páginas: RTMP (nueva) y WHIP.

9 de septiembre de 2024

En [Using EventBridge with IVS Real-Time Streaming](#), hemos agregado un evento de actualización de fases de IVS: error de publicación del participante.

En [Service Quotas](#), hemos agregado valores de TPS para las cinco nuevas operaciones de API y 1 nueva cuota de IngestConfiguration (en “Otras cuotas”).

Los cambios en la API se describen en la tabla de [referencia de la API](#).

[Optimizaciones de transmisión en tiempo real](#)

Se han hecho varias actualizaciones relacionadas con la transmisión simultánea y se ha agregado “Resolución de las capas”.

22 de agosto de 2024

[Publicación o suscripción en la consola](#)

En Introducción a Transmisión en tiempo real de IVS, hemos agregado la publicación y la suscripción en la consola en [Publicación de videos y suscripción para recibirlos](#).

19 de agosto de 2024

[SDK de transmisión: versión 1.15.0 para web](#)

Se han actualizado el número de versión y los enlaces de los artefactos en la [página de inicio de la documentación de IVS](#) y en la guía del SDK de transmisión de baja latencia: [web](#). Consulte también [Notas de la versión](#).

15 de agosto de 2024

También hemos agregado una nueva sección, [Configuración de la suscripción a participantes](#), a la Guía sobre el SDK de transmisión para web.

En Optimizaciones de transmisión, hemos agregado una nueva sección: [Cambio en el valor de MinDelay del búfer de fluctuación del suscriptor](#).

Esto incluye información sobre los SDK de transmisión para web, Android y iOS.

[SDK de transmisión:
versión 1.21.0 para Android y
versión 1.21.0 para iOS](#)

Se han actualizado el número de versión y los enlaces de los artefactos en la [página de inicio de la documentación de IVS](#) y en las guías del SDK de transmisión en tiempo real: [Android](#) y [iOS](#). Consulte también [Notas de la versión](#).

15 de agosto de 2024

También hemos agregado una nueva sección, “Configuración de la suscripción a participantes”, a las guías del SDK de transmisión para [Android](#) y [iOS](#).

[Aclaración de la grabación](#)

Se ha agregado una nota sobre el uso de un bucket de S3 existente en Grabación de participantes individuales (en [1: Crear un depósito de S3](#)) y en Grabación compuesta (en [Requisitos previos](#), paso 3). La configuración Propiedad del objeto debe ser Aplicada al propietario del bucket o Propietario del bucket preferido.

13 de agosto de 2024

[SDK de transmisión:
versión 1.14.0 para web](#)

Se han actualizado el número de versión y los enlaces de los artefactos en la [página de inicio de la documentación de IVS](#) y en la guía del SDK de transmisión de baja latencia: [web](#). Consulte también [Notas de la versión](#).

18 de julio de 2024

SDK de transmisión: versión 1.20.0 para Android y versión 1.20.0 para iOS	Se han actualizado el número de versión y los enlaces de los artefactos en la página de inicio de la documentación de IVS y en las guías del SDK de transmisión en tiempo real: Android y iOS . Consulte también Notas de la versión .	18 de julio de 2024
Introducción a la transmisión en tiempo real	Se ha agregado información sobre los atributos a “Distribución de los tokens de participante”, tanto en “Esquema del token: carga útil” como en “Creación de tokens con la API de transmisión en tiempo real de IVS”.	12 de julio de 2024
Service Quotas	Se ha aumentado el número máximo de suscripciones de fases de 10 000 a 25 000.	27 de junio de 2024
Generación de tokens de participantes con un par de claves	En Introducción a Transmisión en tiempo real de IVS, hemos actualizado Distribución de tokens de participante para explicar las dos formas de generar tokens (API y par de claves) y hemos agregado “Creación de tokens con un par de claves”.	26 de junio de 2024

[Grabación de participantes individuales](#)

Se ha agregado una nueva sección de documentación sobre la [grabación](#), con subdocumentos sobre la [grabación de participantes individuales](#) (nueva) y la grabación compuesta (preexistente). También hemos agregado eventos y ejemplos de cambio de estado de grabación de participantes a [Uso de EventBridge con IVS](#).

20 de junio de 2024

Los cambios en la API se describen en la tabla de [referencia de la API](#).

[Service Quotas](#)

Hemos aumentado la cuota de fases de 100 a 1000.

14 de junio de 2024

[SDK de transmisión: versión 1.19.0 para Android y versión 1.19.0 para iOS](#)

Se han actualizado el número de versión y los enlaces de los artefactos en la [página de inicio de la documentación de IVS](#) y en las guías del SDK de transmisión en tiempo real: [Android](#) y [iOS](#). Consulte también [Notas de la versión](#).

13 de junio de 2024

[SDK de transmisión:
versión 1.13.0 para web](#)

Se han actualizado el número de versión y los enlaces de los artefactos en la [página de inicio de la documentación de IVS](#) y en la guía del SDK de transmisión de baja latencia: [web](#). Consulte también [Notas de la versión](#).

13 de junio de 2024

En la guía, hemos actualizado la información de [Control de errores](#) para el nuevo evento de fase ERROR.

[SDK de transmisión:
versión 1.12.0 para web](#)

Se han actualizado el número de versión y los enlaces de los artefactos en la [página de inicio de la documentación de IVS](#) y en la guía del SDK de transmisión de baja latencia: [web](#). Consulte también [Notas de la versión](#).

20 de mayo de 2024

En la guía, hemos actualizado la información sobre el estado ERRORED de la conexión de fases en la sección [Gestión de los problemas de red](#).

[Optimizaciones de transmisión
en tiempo real](#)

En [Capas, calidades y velocidades de fotogramas predeterminadas](#), se ha modificado la velocidad de bits máxima para dispositivos móviles de capa baja de 150 000 a 100 000 bps.

16 de mayo de 2024

SDK de transmisión: versión 1.18.0 para Android y versión 1.18.0 para iOS	Se han actualizado el número de versión y los enlaces de los artefactos en la página de inicio de la documentación de IVS y en las guías del SDK de transmisión en tiempo real: Android y iOS . Consulte también Notas de la versión .	16 de mayo de 2024
SDK de transmisión: versión 1.11.0 para web	Se han actualizado el número de versión y los enlaces de los artefactos en la página de inicio de la documentación de IVS y en la guía del SDK de transmisión en tiempo real: web . Consulte también Notas de la versión .	6 de mayo de 2024
SDK de transmisión: versión 1.10.1 para web	Se han actualizado el número de versión y los enlaces de los artefactos en la página de inicio de la documentación de IVS y en la guía del SDK de transmisión en tiempo real: web . Consulte también Notas de la versión .	30 de abril de 2024
SDK de transmisión: versión 1.15.2 para Android y versión 1.15.2 para iOS	Se han actualizado el número de versión y los enlaces de los artefactos en la página de inicio de la documentación de IVS y en las guías del SDK de transmisión en tiempo real: Android y iOS . Consulte también Notas de la versión .	30 de abril de 2024

[SDK de transmisión: guía de iOS](#)

En [Publicación de una transmisión de contenido multimedia](#), hemos actualizado el ejemplo de código.

26 de abril de 2024

[SDK de transmisión: versión 1.17.0 para Android y versión 1.17.0 para iOS](#)

Se actualizaron el número de versión y los enlaces de los artefactos para la nueva versión en la guía del SDK de transmisión de transmisión en tiempo real: [Android](#) e [iOS](#). En la [página de destino de la documentación de Amazon IVS](#), se actualizaron los enlaces de referencia del SDK de transmisión para señalar la nueva versión. Consulte también las [Notas de la versión](#) de Amazon IVS para esta versión.

22 de abril de 2024

[Composición del servidor](#)

En [SSC](#), hemos hecho varios cambios, especialmente en el apartado “Diseño”, para explicar los diseños de imagen en imagen y de cuadrícula.

26 de marzo de 2024

En la Guía sobre el SDK de transmisión para web, se ha agregado [Compatibilidad con la representación del servidor](#).

[Compatibilidad con OBS y WHIP](#)

Se ha agregado una nota sobre los problemas de calidad (como la congelación intermitente del video) que pueden producirse con WHIP en OBS.

22 de marzo de 2024

[SDK de transmisión: versión 1.16.0 para Android, versión 1.16.0 para iOS y versión 1.10.0 para web](#)

Se han actualizado el número de versión y los enlaces de los artefactos para la nueva versión en las guías del SDK de transmisión en tiempo real: [Android](#), [iOS](#) y [web](#). En la [página de destino de la documentación de Amazon IVS](#), se actualizaron los enlaces de referencia del SDK de transmisión para señalar la nueva versión. Consulte también las [Notas de la versión](#) de Amazon IVS para esta versión.

21 de marzo de 2024

[SDK de transmisión:
versión 1.15.1 para Android y
versión 1.15.1 para iOS](#)

Se actualizaron el número de versión y los enlaces de los artefactos para la nueva versión en la guía del SDK de transmisión de transmisión en tiempo real: [Android](#) e [iOS](#). En la [página de destino de la documentación de Amazon IVS](#), se actualizaron los enlaces de referencia del SDK de transmisión para señalar la nueva versión. Consulte también las [Notas de la versión](#) de Amazon IVS para esta versión.

13 de marzo de 2024

[SDK de transmisión: modos
de audio móvil](#)

En “Modo preestablecido del audio”, se ha agregado información sobre la categoría de ajustes preestablecidos de control de volumen y un problema conocido de iOS con el ajuste preestablecido de chat de video. En “Casos de uso avanzados”, se ha agregado una nota sobre cómo evitar configuraciones incorrectas y se han agregado secciones sobre “Cancelación de eco de iOS” y “Orígenes de audio personalizados de iOS”.

1 de marzo de 2024

[SDK de transmisión:](#)
[versión 1.15.0 para Android,](#)
[versión 1.15.0 para iOS y](#)
[versión 1.9.0 para web](#)

Se han actualizado el número de versión y los enlaces de los artefactos para la nueva versión en las guías del SDK de transmisión en tiempo real: [Android](#), [iOS](#) y [web](#). En la [página de destino de la documentación de Amazon IVS](#), se actualizaron los enlaces de referencia del SDK de transmisión para señalar la nueva versión. Consulte también las [Notas de la versión](#) de Amazon IVS para esta versión.

22 de febrero de 2024

[Compatibilidad con OBS y WHIP](#)

Se ha agregado una nueva página. En este documento se explica cómo utilizar codificadores compatibles con WHIP, como OBS, para publicar transmisiones en tiempo real de IVS. WHIP (protocolo de ingesta WebRTC-HTTP) es un borrador de IETF desarrollado para estandarizar la ingesta de WebRTC.

6 de febrero de 2024

[SDK de transmisión:](#)
[versión 1.14.1 para Android,](#)
[versión 1.14.1 para iOS y](#)
[versión 1.8.0 para web](#)

Se han actualizado el número de versión y los enlaces de los artefactos para la nueva versión en las guías del SDK de transmisión en tiempo real: [Android](#), [iOS](#) y [web](#). En la [página de destino de la documentación de Amazon IVS](#), se actualizaron los enlaces de referencia del SDK de transmisión para señalar la nueva versión. Consulte también las [Notas de la versión](#) de Amazon IVS para esta versión.

1 de febrero de 2024

En el caso de la guía de Android, hemos agregado un nuevo problema conocido (un tamaño de video inferior a 176 x 176).

En el caso de la guía web, hemos agregado un nuevo problema conocido. La solución alternativa consiste en limitar la resolución de video a 720p al invocar `getUserMedia` o `getDisplayMedia`.

En la sección Optimizaciones de transmisión en tiempo real, hemos actualizado [Configuración de la codificación en capas con transmisión simultánea](#); ahora está

deshabilitada de forma predeterminada.

[SDK de transmisión: versión 1.13.4 para Android, versión 1.13.4 para iOS y versión 1.7.0 para web](#)

Se han actualizado el número de versión y los enlaces de los artefactos para la nueva versión en las guías del SDK de transmisión en tiempo real: [Android](#), [iOS](#) y [web](#). En la [página de destino de la documentación de Amazon IVS](#), se actualizaron los enlaces de referencia del SDK de transmisión para señalar la nueva versión. Consulte también las [Notas de la versión](#) de Amazon IVS para esta versión.

3 de enero de 2024

[Glosario de IVS](#)

Se amplió el glosario para incluir los términos de transmisiones en tiempo real o de baja latencia y chat de IVS.

20 de diciembre de 2023

[Estado de los escenarios: nuevas métricas de CloudWatch](#)

Se cambió el nombre de la métrica PacketLoss (Stage) a DownloadPacketLoss (Stage) y se publicaron métricas adicionales de CloudWatch para la transmisión en tiempo real de IVS:

7 de diciembre de 2023

- DownloadPacketLoss (Stage,Participant)
- DroppedFrames (Stage,Participant)
- SubscribeBitrate (Stage,Participant,MediaType)

Consulte [Monitoreo de transmisión en tiempo real de IVS](#).

[Políticas administradas de IAM](#)

Se agregaron dos políticas administradas: IVSreadOnlyAccess e IVSFullAccess. Consulte:

5 de diciembre de 2023

- La nueva sección sobre [políticas administradas para Amazon IVS](#) en la página Seguridad.
- Cambios en [Paso 3: configurar permisos de IAM](#) en Introducción al streaming de baja latencia de IVS.

[SDK de transmisión: Android 1.13.2 e iOS 1.13.2](#)

Se actualizaron el número de versión y los enlaces de los artefactos para la nueva versión en la guía del SDK de transmisión de transmisión en tiempo real: [Android](#) e [iOS](#).

4 de diciembre de 2023

En la [página de destino de la documentación de Amazon IVS](#), se actualizaron los enlaces de referencia del SDK de transmisión para señalar la nueva versión.

Consulte también las [Notas de la versión](#) de Amazon IVS para esta versión.

[SDK de transmisión: Android 1.13.1](#)

Se actualizaron el número de versión y los enlaces de los artefactos para la nueva versión en la guía del SDK de transmisión en tiempo real: [Android](#).

21 de noviembre de 2023

En la [página de destino de la documentación de Amazon IVS](#), se actualizaron los enlaces de referencia del SDK de transmisión para señalar la nueva versión.

Consulte también las [Notas de la versión](#) de Amazon IVS para esta versión.

[Service Quotas](#)

Cambiamos la “Resolución de publicación de los participantes” de 1080p a 720p.

18 de noviembre de 2023

[SDK de transmisión: Android 1.13.0 e iOS 1.13.0](#)

Se actualizaron el número de versión y los enlaces de los artefactos para la nueva versión en la guía del SDK de transmisión de transmisión en tiempo real: [Android](#) e [iOS](#).

17 de noviembre de 2023

En la [página de destino de la documentación de Amazon IVS](#), se actualizaron los enlaces de referencia del SDK de transmisión para señalar la nueva versión.

Consulte también las [Notas de la versión](#) de Amazon IVS para esta versión.

También hemos realizado varias actualizaciones en las [Optimizaciones de transmisión](#). Entre otras cosas, la característica “Transmisión adaptativa: codificación por capas con transmisión simultánea” ahora requiere una suscripción explícita y solo es compatible con las versiones recientes del SDK.

Grabación compuesta

Se realizaron los siguientes cambios:

16 de noviembre de 2023

- Se agregó una página de [Grabación compuesta](#) para esta nueva característica.
- Se actualizó la política de [Introducción a transmisión en tiempo real de IVS](#) con puntos de conexión S3 en la sección “Configurar los permisos de IAM”.
- [Service Quotas](#) actualizadas con cuotas de tarifas de llamada para los nuevos puntos de conexión.

[Composición del servidor \(SSC\)](#)

16 de noviembre de 2023

La composición del servidor de IVS permite a los clientes transferir la composición y la transmisión de un escenario de IVS a un servicio gestionado por IVS. Las transmisiones SSC y RTMP a un canal se invocan a través de los puntos de conexión del plano de control del IVS en la región de origen del escenario. Consulte:

- [Primeros pasos](#): en la sección “Configurar los permisos de IAM”, añadimos los puntos de conexión SSC a la política.
- [Uso de Amazon EventBridge con IVS](#): hemos añadido nuevas métricas.
- [Composición del servidor](#): este nuevo documento incluye una descripción general e instrucciones de configuración.
- [Service Quotas](#): agregamos nuevos límites de tarifas de llamadas y otras cuotas.

Consulte también:

- Cambios enumerados a continuación en [Cambios de referencia de la API de](#)

[transmisión en tiempo real de IVS](#).

- Cambios enumerados en el [Historial del documento \(transmisión de baja latencia\)](#).

[SDK de transmisión de IVS](#)

En la [Descripción general del SDK de transmisión](#), actualizamos los Requisitos de plataforma > Plataformas nativas para aclarar qué versiones del SDK son compatibles y añadimos “Navegadores móviles (iOS y Android)”.

9 de noviembre de 2023

En la [Guía web de transmisión](#), añadimos “Limitaciones de la web móvil”.

[SDK de transmisión de IVS](#)

Hemos añadido una nueva página sobre [Filtros de cámara de terceros](#).

9 de noviembre de 2023

[Introducción a transmisión en tiempo real de IVS](#)

Hemos actualizado los procedimientos en [Configurar permisos de IAM](#).

20 de octubre de 2023

[Monitoreo de transmisión en tiempo real](#)

En [Métricas de CloudWatch: transmisión en tiempo real de IVS](#), agregamos valores de muestra para las dimensiones.

17 de octubre de 2023

[SDK de transmisión: guía para la web](#)

Hemos realizado varios cambios en el [Monitoreo del estado de sonido desactivado en los contenidos multimedia del participante remoto](#).

17 de octubre de 2023

[SDK de transmisión: web 1.6.0](#)

Se actualizaron el número de versión y los enlaces de los artefactos para la nueva versión en la guía del SDK de transmisión en tiempo real: [Web](#).

16 de octubre de 2023

La [página de inicio de la documentación de Amazon IVS](#) siempre indica la versión reciente de las Referencias del SDK del Transmisor.

Consulte también las [Notas de la versión](#) de Amazon IVS para esta versión.

En la Guía Web, en “Recuperar un MediaStream de un dispositivo”, también eliminamos las dos líneas max; la mejor práctica es especificar únicamente `ideal`.

En Optimizaciones de transmisión en tiempo real, agregamos una nueva sección, [Optimización de la tasa de bits de audio y soporte estéreo](#).

[Estado de los escenarios: nuevas métricas de CloudWatch](#)

Se publicaron métricas de CloudWatch para transmisión en tiempo real de IVS. Consulte [Monitoreo de transmisión en tiempo real de IVS](#).

12 de octubre de 2023

[SDK de transmisión: Android 1.12.1](#)

Se actualizaron el número de versión y los enlaces de los artefactos para la nueva versión en la guía del SDK de transmisión en tiempo real: [Android](#). También se agregó una nueva sección, [Uso de micrófonos Bluetooth](#).

12 de octubre de 2023

La [página de inicio de la documentación de Amazon IVS](#) siempre indica la versión reciente de las Referencias del SDK del Transmisor.

Consulte también las [Notas de la versión](#) de Amazon IVS para esta versión.

SDK de transmisión: web 1.5.2	<p>Se actualizaron el número de versión y los enlaces de los artefactos para la nueva versión en la guía del SDK de transmisión en tiempo real: Web.</p> <p>La página de inicio de la documentación de Amazon IVS siempre indica la versión reciente de las Referencias del SDK del Transmisor.</p> <p>Consulte también las Notas de la versión de Amazon IVS para esta versión.</p>	14 de septiembre de 2023
Introducción a transmisión en tiempo real de IVS	En Android > Instalar el SDK de transmisión , se agregó el enlace de datos.	12 de septiembre de 2023
Control de errores del SDK de transmisión	Se agregaron secciones de “Gestión de errores” a las guías del SDK de transmisión: Web , Android e iOS .	12 de septiembre de 2023
Introducción a transmisión en tiempo real de IVS	En Distribuir los tokens de participante , se agregó una nota Importante sobre no crear funciones basadas en el formato de token actual.	1 de septiembre de 2023
Introducción a transmisión en tiempo real de IVS	En Configurar los permisos de IAM , se actualizó el conjunto de permisos.	31 de agosto de 2023

[SDK de transmisión:](#)
[Web 1.5.1, Android 1.12.0 y](#)
[iOS 1.12.0](#)

Se actualizaron el número de versión y los enlaces de los artefactos para la nueva versión en las guías del SDK de transmisión en tiempo real: [Web](#), [Android](#) e [iOS](#).

23 de agosto de 2023

En la [página de destino de la documentación de Amazon IVS](#), se actualizaron los enlaces de referencia del SDK de transmisión para señalar la nueva versión.

Consulte también las [Notas de la versión](#) de Amazon IVS para esta versión.

[Lanzamiento de transmisión en tiempo real](#)

Esta versión incluye cambios importantes en la documentación. Cambiamos el nombre de la documentación anterior por transmisión en tiempo real de baja latencia de IVS y publicamos la nueva documentación de transmisión en tiempo real de IVS. La [página de destino de la documentación de Amazon IVS](#) ahora tiene secciones diferentes para la transmisión en tiempo real y la transmisión de baja latencia. Cada sección tiene su propia guía del usuario y referencia de API.

Para ver otros cambios en la documentación, consulte [Historial del documento \(transmisión de baja latencia\)](#).

7 de agosto de 2023

[SDK de transmisión:](#)
[Web 1.5.0, Android 1.11.0 y](#)
[iOS 1.11.0](#)

Se actualizaron el número de versión y los enlaces de los artefactos para la nueva versión en las guías del SDK de transmisión: [Web](#), [Android](#) e [iOS](#).

7 de agosto de 2023

En la [página de destino de la documentación de Amazon IVS](#), se actualizaron los enlaces de referencia del SDK de transmisión para señalar la nueva versión.

Consulte también las [Notas de la versión](#) de Amazon IVS para esta versión.

Cambios en la Referencia de la API de transmisión en tiempo real de IVS

Cambio de la API	Descripción	Fecha
Actualización de los objetos Event y Video	<p>En el objeto Event, hemos agregado más valores válidos para <code>errorCode</code> .</p> <p>En el objeto Video, hemos aclarado que <code>height</code> y <code>width</code> deben ser números pares.</p>	2 de octubre de 2024
Ingesta RTMP	Hemos agregado dos objetos: <code>IngestConfiguration</code> e <code>IngestConfigurationSummary</code> . Hemos agregado cinco puntos de conexión de <code>IngestConfiguration</code> (<code>Create</code> , <code>Delete</code> , <code>Get</code> , <code>List</code> y <code>Update</code>).	9 de septiembre de 2024

Cambio de la API	Descripción	Fecha
	<p>Hemos actualizado DeleteStage (descripción de la operación) y DisconnectParticipant (descripciones de la operación y participantId).</p> <p>Hemos modificado el objeto Participant (hemos agregado el campo protocol); eso afecta a la respuesta de GetParticipant.</p> <p>Hemos modificado el objeto StageEndpoints (hemos agregado los campos rtmp y rtmps); eso afecta a las respuestas de CreateStage, GetStage y UpdateStage. También hemos actualizado la descripción de este objeto (hemos agregado una recomendación de almacenamiento en caché).</p>	
Generación de tokens de participantes con un par de claves	Hemos agregado tres objetos (PublicKey, PublicKey Summary y StageEndpoints) y cuatro puntos de conexión (DeletePublicKey, GetPublicKey, ImportPublicKey y ListPublicKeys). Hemos modificado el objeto Stage (hemos agregado el campo endpoints); eso afecta a las respuestas de CreateStage, GetStage y UpdateStage.	26 de junio de 2024
Grabación de participantes individuales	Hemos agregado un objeto (AutoParticipantRecordingConfiguration) y hemos modificado tres objetos (Participant, ParticipantSummary y Stage). Esto afecta a cinco puntos de conexión: solicitud y respuesta de CreateStage, respuesta de GetParticipant, respuesta de GetStage, solicitud y respuesta de ListParticipants y solicitud y respuesta de UpdateStage.	20 de junio de 2024

Cambio de la API	Descripción	Fecha
Eliminación de svcs de los patrones de ARN	Los patrones de ARN que especificaron [is]vs se actualizaron para especificar ivs. Esto afecta a los tres puntos de conexión de etiqueta y al campo <code>ChannelDestinationConfiguration\$channelArn</code> .	25 de abril de 2024
Actualizaciones de composición del servidor	Hemos agregado un objeto: <code>PipConfiguration</code> . Hemos modificado dos objetos (<code>LayoutConfiguration</code> y <code>GridConfiguration</code>). Esto afecta a la respuesta de <code>GetComposition</code> y a la solicitud y respuesta de <code>StartComposition</code> .	13 de marzo de 2024
Grabación compuesta	Hemos agregado 4 puntos de conexión de <code>StorageConfiguration</code> y 7 objetos (<code>DestinationDetail</code> , <code>RecordingConfiguration</code> , <code>S3DestinationConfiguration</code> , <code>S3Detail</code> , <code>S3StorageConfiguration</code> , <code>StorageConfiguration</code> , <code>StorageConfiguration Summary</code>). Modificamos 3 objetos (<code>Composition</code> , <code>Destination</code> y <code>DestinationConfiguration</code>). Esto afecta a la respuesta de <code>GetComposition</code> y a la solicitud y respuesta de <code>StartComposition</code> .	16 de noviembre de 2023
Composición del servidor	Hemos añadido 8 puntos de conexión de <code>Composition</code> y <code>EncoderConfiguration</code> y 11 objetos (<code>ChannelDestinationConfiguration</code> , <code>Composition</code> , <code>CompositionSummary</code> , <code>Destination</code> , <code>DestinationConfiguration</code> , <code>EncoderConfiguration</code> , <code>EncoderConfigurationSummary</code> , <code>GridConfiguration</code> , <code>LayoutConfiguration</code> y <code>Video</code>).	16 de noviembre de 2023
Estado de los escenarios: datos sobre nuevos participantes	Se agregaron seis campos al objeto participante : <code>browserName</code> , <code>browserVersion</code> , <code>ispName</code> , <code>osName</code> , <code>osVersion</code> y <code>sdkVersion</code> . Esto afecta a la respuesta de <code>GetParticipant</code> .	12 de octubre de 2023

Cambio de la API	Descripción	Fecha
Token de participante	Se agregó una nota Importante sobre no crear funciones basadas en el formato de token actual.	1 de septiembre de 2023
Lanzamiento de transmisión en tiempo real de IVS	<p>Esta versión incluye cambios importantes en la documentación. Cambiamos el nombre de la documentación anterior por transmisión en tiempo real de baja latencia de IVS y publicamos la nueva documentación de transmisión en tiempo real de IVS. La página de destino de la documentación de Amazon IVS ahora tiene secciones diferentes para la transmisión en tiempo real y la transmisión de baja latencia. Cada sección tiene su propia guía del usuario y referencia de API.</p> <p>La Referencia de la API de transmisión en tiempo real de IVS forma parte de la documentación de transmisión en tiempo real de IVS. Anteriormente, se titulaba Referencia de la API de escenarios de IVS. Su historial anterior se describe en Historial del documento (transmisión de baja latencia).</p>	7 de agosto de 2023

Notas de la versión de IVS | Transmisión en tiempo real

Este documento contiene todas las notas de la versión de Transmisión en tiempo real de Amazon IVS, empezando por la última, organizadas por fecha de publicación.

10 de octubre de 2024

SDK de transmisión de IVS: versión 1.17.0 para web (transmisión en tiempo real)

Plataforma	Descargas y cambios
SDK de transmisión para web (versión 1.17.0)	<p>Documentación de referencia: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference</p> <ul style="list-style-type: none">• Correcciones de errores menores.

10 de octubre de 2024

SDK de transmisión de Amazon IVS: versión 1.23.0 para Android y versión 1.23.0 para iOS (transmisión en tiempo real)

Plataforma	Descargas y cambios
SDK de transmisión para Android (versión 1.23.0)	<p>Documentación de referencia: https://aws.github.io/amazon-ivs-broadcast-docs/1.23.0/android/</p> <ul style="list-style-type: none">• Con esta versión, también comenzamos a publicar una versión del SDK de transmisión para Android que incluye símbolos de depuración. Consulte Uso del SDK con símbolos de depuración.• Correcciones de errores menores.

Plataforma	Descargas y cambios
SDK de transmisión para iOS (versión 1.23.0)	<p>Descarga para transmisión en tiempo real: https://broadcast.live-video.net/1.23.0/AmazonIVSBroadcast-Stages.xcframework.zip</p> <p>Documentación de referencia: https://aws.github.io/amazon-ivs-broadcast-docs/1.23.0/ios/</p> <ul style="list-style-type: none"> • Correcciones de errores menores.

Tamaño del SDK de transmisión: Android

Arquitectura	Tamaño comprimido	Tamaño sin comprimir
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

Tamaño del SDK de transmisión: iOS

Arquitectura	Tamaño comprimido	Tamaño sin comprimir
arm64	3,542 MB	8,316 MB

11 de septiembre de 2024

SDK de transmisión de Amazon IVS: versión 1.22.0 para Android y versión 1.22.0 para iOS (transmisión en tiempo real)

Plataforma	Descargas y cambios
SDK de transmisión para Android (versión 1.22.0)	<p>Documentación de referencia: https://aws.github.io/amazon-ivs-broadcast-docs/1.22.0/android/</p> <ul style="list-style-type: none"> Se ha corregido un error que provocaba que algunos dispositivos Android mostraran un marco negro en la vista previa tras cambiar las entradas de la cámara. Correcciones de errores menores.
SDK de transmisión para iOS (versión 1.22.0)	<p>Descarga para transmisión en tiempo real: https://broadcast.live-video.net/1.22.0/AmazonIVSBroadcast-Stages.xcframework.zip</p> <p>Documentación de referencia: https://aws.github.io/amazon-ivs-broadcast-docs/1.22.0/ios/</p> <ul style="list-style-type: none"> Correcciones de errores menores.

Tamaño del SDK de transmisión: Android

Arquitectura	Tamaño comprimido	Tamaño sin comprimir
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

Tamaño del SDK de transmisión: iOS

Arquitectura	Tamaño comprimido	Tamaño sin comprimir
arm64	3,488 MB	8,199 MB

11 de septiembre de 2024

SDK de transmisión de IVS: versión 1.16.0 para web (transmisión en tiempo real)

Plataforma	Descargas y cambios
SDK de transmisión para web (versión 1.16.0)	<p>Documentación de referencia: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference</p> <ul style="list-style-type: none"> Correcciones de errores menores.

9 de septiembre de 2024

Ingesta RTMP

Como alternativa al uso del SDK de transmisión de IVS, ahora puede publicar videos en una fase de IVS desde un origen RTMP (además de WHIP, que ya era compatible). Para ver los cambios en la documentación, consulte [Historial de documentos](#) (tanto en la Guía del usuario como en las tablas de referencia de la API).

19 de agosto de 2024

Publicación o suscripción en la consola

Ahora puede publicar y suscribirse desde la consola de IVS. En [Introducción a Transmisión en tiempo real de IVS](#), consulte [Publicación de videos y suscripción para recibirlos](#).

15 de agosto de 2024

SDK de transmisión de IVS: versión 1.15.0 para web (transmisión en tiempo real)

Plataforma	Descargas y cambios
SDK de transmisión para web (versión 1.15.0)	<p>Documentación de referencia: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference</p> <ul style="list-style-type: none">• Se ha corregido una condición de carrera que afectaba a la calidad del contenido multimedia del publicador cuando se llamaba a <code>join()</code> repetidamente. Llamar a <code>join()</code> sucesivamente ya no vuelve a desencadenar el evento <code>STAGE_PARTICIPANT_JOINED</code>, junto con los consiguientes cambios de estado de publicación y transmisión.• Se ha corregido un error que provocaba problemas al analizar los tokens de participantes cuando se utilizaban caracteres no textuales en el campo <code>attributes</code> del token.• Se ha agregado un método para configurar los suscriptores de un participante. Inicialmente, solo se puede configurar el retardo mínimo del búfer de fluctuación. Consulte la documentación de referencia del SDK, Configuración de la suscripción a participantes en la Guía sobre el SDK de transmisión para web y Cambio en el valor de <code>MinDelay</code> del búfer de fluctuación del suscriptor en Optimizaciones de transmisión.

15 de agosto de 2024

SDK de transmisión de Amazon IVS: versión 1.21.0 para Android y versión 1.21.0 para iOS (transmisión en tiempo real)

Plataforma	Descargas y cambios
<p>SDK de transmisión para Android (versión 1.21.0)</p>	<p>Documentación de referencia: https://aws.github.io/amazon-ivs-broadcast-docs/1.21.0/android/</p> <ul style="list-style-type: none"> • Se ha corregido un error que afectaba a los dispositivos con chipsets MT6765 que hacía que la vista previa para suscriptores mostrara marcos negros en algunas circunstancias. • Se ha agregado un método para configurar los suscriptores de un participante. Inicialmente, solo se puede configurar el retardo mínimo del búfer de fluctuación. Consulte la documentación de referencia del SDK, Configuración de la suscripción a participantes en la Guía sobre el SDK de transmisión para Android y Cambio en el valor de MinDelay del búfer de fluctuación del suscriptor en Optimizaciones de transmisión. • Correcciones de errores menores.
<p>SDK de transmisión para iOS (versión 1.21.0)</p>	<p>Descarga para transmisión en tiempo real: https://broadcast.live-video.net/1.21.0/AmazonIVSBroadcast-Stages.xcframework.zip</p> <p>Documentación de referencia: https://aws.github.io/amazon-ivs-broadcast-docs/1.21.0/ios/</p> <ul style="list-style-type: none"> • Se ha agregado un método para configurar los suscriptores de un participante. Inicialmente, solo se puede configurar el retardo

Plataforma	Descargas y cambios
	<p>mínimo del búfer de fluctuación. Consulte la documentación de referencia del SDK, Configuración de la suscripción a participantes en la Guía sobre el SDK de transmisión para iOS y Cambio en el valor de MinDelay del búfer de fluctuación del suscriptor en Optimizaciones de transmisión.</p> <ul style="list-style-type: none"> • Correcciones de errores menores.

Tamaño del SDK de transmisión: Android

Arquitectura	Tamaño comprimido	Tamaño sin comprimir
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

Tamaño del SDK de transmisión: iOS

Arquitectura	Tamaño comprimido	Tamaño sin comprimir
arm64	3,485 MB	8,199 MB

18 de julio de 2024

SDK de transmisión de IVS: versión 1.14.0 para web (transmisión en tiempo real)

Plataforma	Descargas y cambios
SDK de transmisión para web (versión 1.14.0)	<p>Documentación de referencia: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference</p> <ul style="list-style-type: none"> • Mejoras en la documentación de la API. • Se han corregido los valores atípicos de las estadísticas de video y audio registrados durante el restablecimiento de la conexión. • Actualizaciones de dependencia menores.

18 de julio de 2024

SDK de transmisión de Amazon IVS: versión 1.20.0 para Android y versión 1.20.0 para iOS (transmisión en tiempo real)

Plataforma	Descargas y cambios
SDK de transmisión para Android (versión 1.20.0)	<p>Documentación de referencia: https://aws.github.io/amazon-ivs-broadcast-docs/1.20.0/android</p> <ul style="list-style-type: none"> • Se ha corregido un error que impedía que el SDK de transmisión se pusiera en marcha en Chromebooks con procesadores Intel. • Correcciones de errores menores.
SDK de transmisión para iOS (versión 1.20.0)	<p>Descarga para transmisión en tiempo real: https://broadcast.live-video.net/1.20.0/AmazonIVSBroadcast-Stages.xcframework.zip</p>

Plataforma	Descargas y cambios
	Documentación de referencia: https://aws.github.io/amazon-ivs-broadcast-docs/1.20.0/ios <ul style="list-style-type: none"> • Correcciones de errores menores.

Tamaño del SDK de transmisión: Android

Arquitectura	Tamaño comprimido	Tamaño sin comprimir
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

Tamaño del SDK de transmisión: iOS

Arquitectura	Tamaño comprimido	Tamaño sin comprimir
arm64	3,465 MB	8,164 MB

26 de junio de 2024

Generación de tokens de participantes con un par de claves

Ahora puede generar tokens de participante en su propia aplicación de servidor mediante un par de claves. Esto le permite evitar tener que llamar a `CreateParticipantToken` cada vez que necesite un token de participante. Para ver los cambios en la documentación, consulte [Historial de documentos](#) (tanto en la Guía del usuario como en las tablas de referencia de la API).

20 de junio de 2024

Grabación de participantes individuales

La grabación de participantes individuales permite a los clientes de IVS que transmiten en tiempo real grabar a los publicadores de fases de IVS de forma individual en buckets de S3. Consulte [Grabación](#), [Grabación de participantes individuales](#) y cambios en la [Referencia de la API de transmisión en tiempo real](#). (Para ver otros cambios en la documentación, consulte [Historial de documentos](#)).

13 de junio de 2024

SDK de transmisión de Amazon IVS: versión 1.19.0 para Android y versión 1.19.0 para iOS (transmisión en tiempo real)

Plataforma	Descargas y cambios
SDK de transmisión para Android (versión 1.19.0)	<p>Documentación de referencia: https://aws.github.io/amazon-ivs-broadcast-docs/1.19.0/android</p> <ul style="list-style-type: none"> Las versiones recientes de Android requieren un ícono en la notificación que se muestra al capturar la pantalla. Si lo desea, ahora puede personalizar el ícono llamando a <code>setSmallIcon</code> en el método <code>Notification.Builder</code> devuelto por <code>Session#createServiceNotificationBuilder</code>. Se ha mejorado el tiempo de recuperación de la conexión en los dispositivos que pasan de una conexión wifi a una conexión móvil. Este cambio requiere el permiso <code>CHANGE_NETWORK_STATE</code>.
SDK de transmisión para iOS (versión 1.19.0)	<p>Descarga para transmisión en tiempo real: https://broadcast.live-video.net/1.19.0/AmazonIVSBroadcast-Stages.xcframework.zip</p>

Plataforma	Descargas y cambios
	Documentación de referencia: https://aws.github.io/amazon-ivs-broadcast-docs/1.19.0/ios <ul style="list-style-type: none"> Correcciones de errores menores.

Tamaño del SDK de transmisión: Android

Arquitectura	Tamaño comprimido	Tamaño sin comprimir
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

Tamaño del SDK de transmisión: iOS

Arquitectura	Tamaño comprimido	Tamaño sin comprimir
arm64	3,393 MB	7,949 MB

13 de junio de 2024

SDK de transmisión de IVS: versión 1.13.0 para web (transmisión en tiempo real)

Plataforma	Descargas y cambios
SDK de transmisión para web (versión 1.13.0)	Documentación de referencia: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference

Plataforma	Descargas y cambios
	<ul style="list-style-type: none"> • Se ha actualizado la duración del cambio de comportamiento del evento para <code>StageEvents.STAGE_PARTICIPANT_SUBSCRIBE_STATE_CHANGED</code> y <code>StageEvents.STAGE_PARTICIPANT_PUBLISH_STATE_CHANGED</code>. Los participantes ahora permanecen en el estado <code>ATTEMPTING_SUBSCRIBE</code> o <code>ATTEMPTING_PUBLISH</code> durante más tiempo, hasta que el evento <code>ERRORED</code> se active. • Se ha agregado el evento <code>StageEvents.ERROR</code> para escuchar los errores detectados por el SDK. Consulte Error Handling en Real-Time Broadcast SDK: Web Guide para obtener más información.

20 de mayo de 2024

SDK de transmisión de IVS: versión 1.12.0 para web (transmisión en tiempo real)

Plataforma	Descargas y cambios
SDK de transmisión para web (versión 1.12.0)	<p>Documentación de referencia: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference</p> <ul style="list-style-type: none"> • Se ha mejorado la gestión de los reintentos para las operaciones de publicación y suscripción. • Se han mejorado los análisis, en concreto la medición de la latencia y la calidad del audio.

16 de mayo de 2024

SDK de transmisión de Amazon IVS: versión 1.18.0 para Android y versión 1.18.0 para iOS (transmisión en tiempo real)

Plataforma	Descargas y cambios
SDK de transmisión para Android (versión 1.18.0)	<p>Documentación de referencia: https://aws.github.io/amazon-ivs-broadcast-docs/1.18.0/android</p> <ul style="list-style-type: none"> El SDK ahora envía códigos de error específicos cuando el plano de control de AWS elimina una fase conectada o cuando se revoca el token en uso. Correcciones de errores menores.
SDK de transmisión para iOS (versión 1.18.0)	<p>Descarga para transmisión en tiempo real: https://broadcast.live-video.net/1.18.0/AmazonIVSBroadcast-Stages.xcframework.zip</p> <p>Documentación de referencia: https://aws.github.io/amazon-ivs-broadcast-docs/1.18.0/ios</p> <ul style="list-style-type: none"> El SDK ahora envía códigos de error específicos cuando el plano de control de AWS elimina una fase conectada o cuando se revoca el token en uso. Se ha agregado el método <code>setVideoZoomFactor</code> de <code>IVSCamera</code> y los métodos <code>IVSCameraDelegate</code> asociados.

Tamaño del SDK de transmisión: Android

Arquitectura	Tamaño comprimido	Tamaño sin comprimir
arm64-v8a	5,275 MB	13,279 MB

Arquitectura	Tamaño comprimido	Tamaño sin comprimir
armeabi-v7a	4,573 MB	9,254 MB
x86_64	5,472 MB	14,142 MB
x86	5,664 MB	14,554 MB

Tamaño del SDK de transmisión: iOS

Arquitectura	Tamaño comprimido	Tamaño sin comprimir
arm64	3,393 MB	7,916 MB

6 de mayo de 2024

SDK de transmisión de IVS: versión 1.11.0 para web (transmisión en tiempo real)

Plataforma	Descargas y cambios
SDK de transmisión para web (versión 1.11.0)	<p>Documentación de referencia: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference</p> <ul style="list-style-type: none"> Se ha corregido un caso de borde en el que el SDK no intentaba recuperarse en una fase DISCONNECT . Se ha actualizado el mensaje de error por un error de tiempo de espera <code>join()</code>. En lugar de “InitialConnectTimedOut después de 10 segundos”, el SDK ahora devuelve “Se agotó el tiempo de espera de la operación”.

30 de abril de 2024

SDK de transmisión de IVS: versión 1.10.1 para web (transmisión en tiempo real)

Plataforma	Descargas y cambios
SDK de transmisión para web (versión 1.10.1)	Documentación de referencia: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference <ul style="list-style-type: none"> • Correcciones de errores menores.

30 de abril de 2024

SDK de transmisión de Amazon IVS: versión 1.15.2 para Android y versión 1.15.2 iOS (transmisión en tiempo real)

Plataforma	Descargas y cambios
SDK de transmisión para Android (versión 1.15.2)	Documentación de referencia: https://aws.github.io/amazon-ivs-broadcast-docs/1.15.2/android <ul style="list-style-type: none"> • Correcciones de errores menores. Actualice a esta versión solo si tiene un motivo específico para hacerlo; de lo contrario, utilice la versión más reciente publicada.
SDK de transmisión para iOS (versión 1.15.2)	Descarga para transmisión en tiempo real: https://broadcast.live-video.net/1.15.2/AmazonIVSBroadcast-Stages.xcframework.zip Documentación de referencia: https://aws.github.io/amazon-ivs-broadcast-docs/1.15.2/ios

Plataforma	Descargas y cambios
	<ul style="list-style-type: none"> Correcciones de errores menores. Actualice a esta versión solo si tiene un motivo específico para hacerlo; de lo contrario, utilice la versión más reciente publicada.

Tamaño del SDK de transmisión: Android

Arquitectura	Tamaño comprimido	Tamaño sin comprimir
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

Tamaño del SDK de transmisión: iOS

Arquitectura	Tamaño comprimido	Tamaño sin comprimir
arm64	3,359 MB	7,836 MB

22 de abril de 2024

SDK de transmisión de Amazon IVS: versión 1.17.0 para Android y versión 1.17.0 para iOS (transmisión en tiempo real)

Plataforma	Descargas y cambios
SDK de transmisión para Android (versión 1.17.0)	Documentación de referencia: https://aws.github.io/amazon-ivs-broadcast-docs/1.17.0/android

Plataforma	Descargas y cambios
	<ul style="list-style-type: none"> Se ha corregido un error poco frecuente que podía producirse durante la publicación.
SDK de transmisión para iOS (versión 1.17.0)	<p>Descarga para transmisión en tiempo real: https://broadcast.live-video.net/1.17.0/AmazonIVSBroadcast-Stages.xcframework.zip</p> <p>Documentación de referencia: https://aws.github.io/amazon-ivs-broadcast-docs/1.17.0/ios</p> <ul style="list-style-type: none"> El marco AmazonIVSBroadcast ahora incluye un manifiesto de privacidad, tal y como exige Apple.

Tamaño del SDK de transmisión: Android

Arquitectura	Tamaño comprimido	Tamaño sin comprimir
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

Tamaño del SDK de transmisión: iOS

Arquitectura	Tamaño comprimido	Tamaño sin comprimir
arm64	3,388 MB	7,916 MB

21 de marzo de 2024

SDK de transmisión de Amazon IVS: versión 1.16.0 para Android, versión 1.16.0 para iOS y versión 1.10.0 para web (transmisión en tiempo real)

Plataforma	Descargas y cambios
SDK de transmisión para web (versión 1.10.0)	<p>Documentación de referencia: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference</p> <ul style="list-style-type: none"> Se ha corregido un error intermitente que se producía al limpiar las conexiones después de cancelar la suscripción o salir de una fase.
SDK de transmisión para Android (versión 1.16.0)	<p>Documentación de referencia: https://aws.github.io/amazon-ivs-broadcast-docs/1.16.0/android</p> <ul style="list-style-type: none"> Se ha corregido un bloqueo de las vistas previas en la variante Exynos de los dispositivos Samsung con Android 14. Se ha agregado una función para consultar las capacidades de zoom de la cámara y configurar el factor de zoom.
SDK de transmisión para iOS (versión 1.16.0)	<p>Descarga para transmisión en tiempo real: https://broadcast.live-video.net/1.16.0/AmazonIVSBroadcast-Stages.xcframework.zip</p> <p>Documentación de referencia: https://aws.github.io/amazon-ivs-broadcast-docs/1.16.0/ios</p> <ul style="list-style-type: none"> Correcciones de errores menores.

Tamaño del SDK de transmisión: Android

Arquitectura	Tamaño comprimido	Tamaño sin comprimir
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

Tamaño del SDK de transmisión: iOS

Arquitectura	Tamaño comprimido	Tamaño sin comprimir
arm64	3,361 MB	7,836 MB

13 de marzo de 2024

SDK de transmisión de Amazon IVS: versión 1.15.1 para Android y versión 1.15.1 para iOS (transmisión en tiempo real)

Plataforma	Descargas y cambios
SDK de transmisión para Android (versión 1.15.1)	<p>Documentación de referencia: https://aws.github.io/amazon-ivs-broadcast-docs/1.15.1/android</p> <ul style="list-style-type: none"> Se ha corregido un error poco frecuente que se producía al suscribirse a un participante remoto.
SDK de transmisión para iOS (versión 1.15.1)	<p>Descarga para transmisión en tiempo real: https://broadcast.live-video.net/1.15.1/AmazonIVSBroadcast-Stages.xcframework.zip</p>

Plataforma	Descargas y cambios
	<p>Documentación de referencia: https://aws.github.io/amazon-ivs-broadcast-docs/1.15.1/ios</p> <ul style="list-style-type: none"> Se ha corregido un error poco frecuente que se producía al suscribirse a un participante remoto.

Tamaño del SDK de transmisión: Android

Arquitectura	Tamaño comprimido	Tamaño sin comprimir
arm64-v8a	5243 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

Tamaño del SDK de transmisión: iOS

Arquitectura	Tamaño comprimido	Tamaño sin comprimir
arm64	3,358 MB	7,820 MB

13 de marzo de 2024

Actualizaciones de la API de composición del servidor

Hemos introducido nuevas propiedades en `GridConfiguration` y un nuevo diseño de imagen en imagen, lo que mejora las opciones de personalización de las composiciones. Para ver cambios específicos en la documentación, consulte [Historial de documentos](#) (consulte la tabla de cambios de la referencia de la API).

Importante: Asegúrese de que su aplicación no dependa de las características específicas del diseño actual, como el tamaño y la posición de los mosaicos. Se pueden introducir mejoras visuales en los diseños en cualquier momento.

8 de marzo de 2024

Actualizaciones del diseño de composición del servidor

Hoy hemos habilitado los cambios en el diseño de cuadrícula predeterminado que se describen en la entrada del [7 de febrero de 2024](#).

22 de febrero de 2024

SDK de transmisión de Amazon IVS: versión 1.15.0 para Android, versión 1.15.0 para iOS y versión 1.9.0 para web (transmisión en tiempo real)

Plataforma	Descargas y cambios
SDK de transmisión para web (versión 1.9.0)	<p>Documentación de referencia: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference</p> <ul style="list-style-type: none"> Se ha mejorado la gestión de errores internos.
SDK de transmisión para Android (versión 1.15.0)	<p>Documentación de referencia: https://aws.github.io/amazon-ivs-broadcast-docs/1.15.0/android</p> <ul style="list-style-type: none"> Correcciones de errores menores.
SDK de transmisión para iOS (versión 1.15.0)	<p>Descarga para transmisión en tiempo real: https://broadcast.live-video.net/1.15.0/AmazonIVSBroadcast-Stages.xcframework.zip</p> <p>Documentación de referencia: https://aws.github.io/amazon-ivs-broadcast-docs/1.15.0/ios</p>

Plataforma	Descargas y cambios
	<ul style="list-style-type: none"> • Se ha agregado una extensión <code>AVPictureInPictureController</code> para permitir la creación de una nueva instancia con <code>IVSImagePreviewView</code>. • Se ha agregado una nueva API en <code>IVSImageDevice</code> para crear un objeto <code>AVSampleBufferDisplayLayer</code> en el que se renderice el dispositivo. • Se ha corregido una velocidad de bits baja en los dispositivos con iOS 17 y versiones posteriores. • Correcciones de errores menores.

Tamaño del SDK de transmisión: Android

Arquitectura	Tamaño comprimido	Tamaño sin comprimir
arm64-v8a	5243 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

Tamaño del SDK de transmisión: iOS

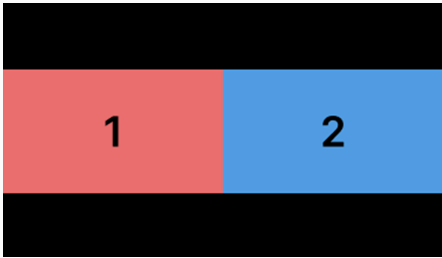
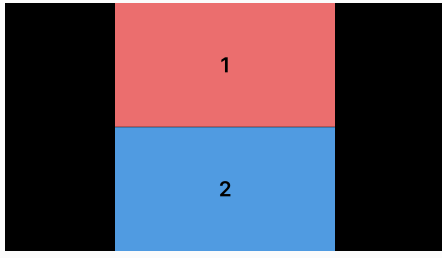
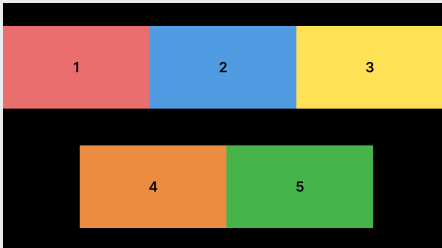

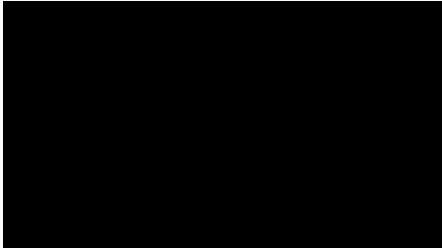
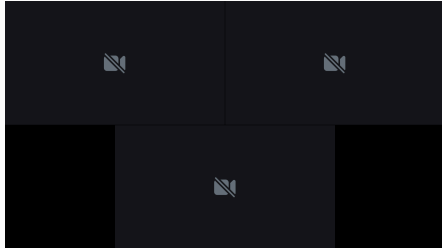
Arquitectura	Tamaño comprimido	Tamaño sin comprimir
arm64	3,358 MB	7,820 MB

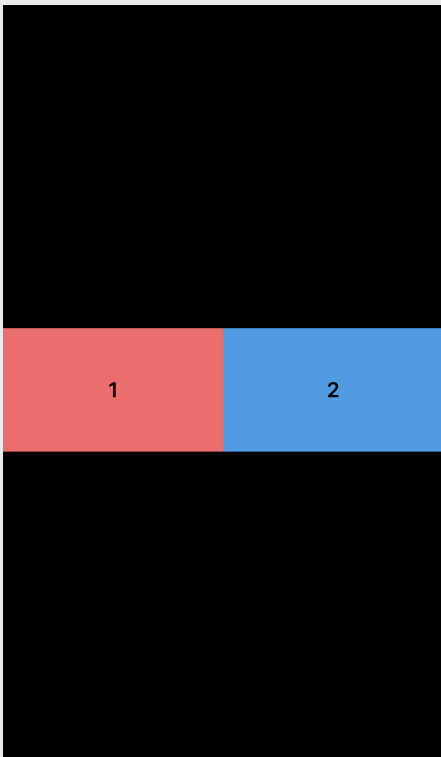
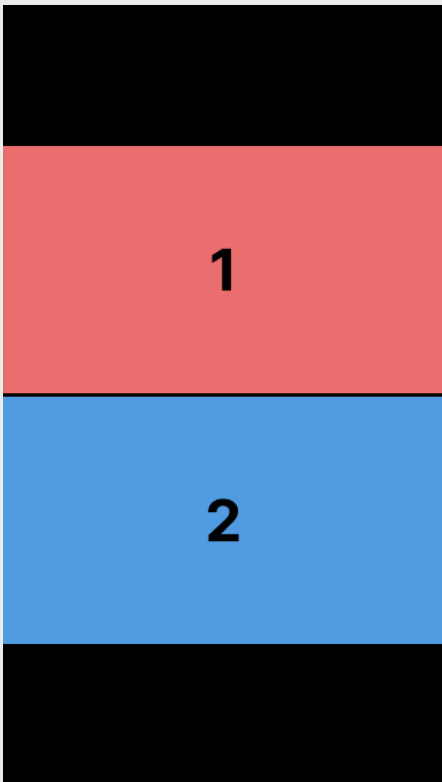
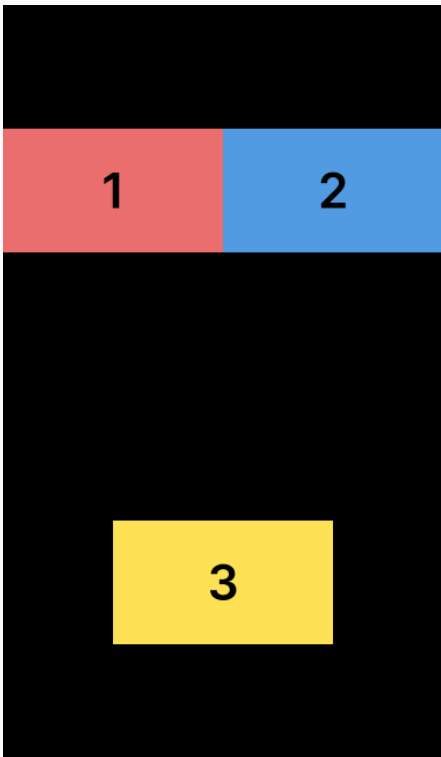
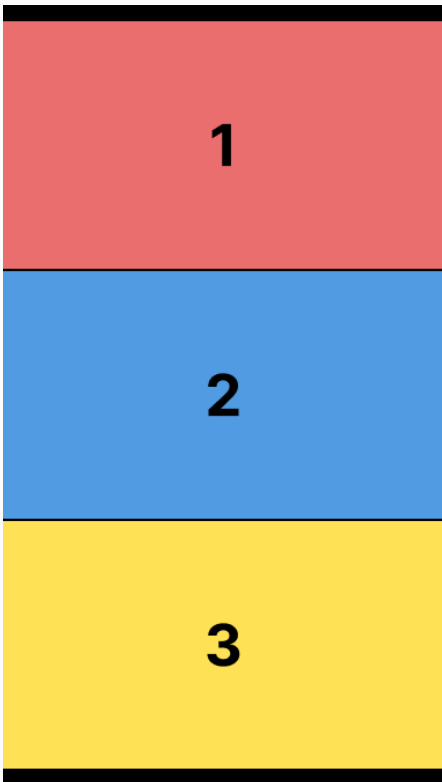
7 de febrero de 2024

Actualizaciones del diseño de composición del servidor

Esta versión introduce mejoras visuales en el diseño de cuadrícula predeterminado. Estos cambios optimizarán la visualización del video y reducirán el espacio en blanco. Estos cambios se habilitarán el 7 de marzo de 2024.

Importante: Asegúrese de que su aplicación no dependa de las características específicas del diseño actual, como el tamaño y la posición de los mosaicos. Se pueden introducir mejoras visuales en los diseños en cualquier momento.

Descripción del cambio	Antiguo	New
<p>Selecciona automáticamente la ubicación óptima de los participantes para maximizar el tamaño del video.</p>	 <p>Diagrama de cuadrícula antiguo con dos mosaicos horizontales (rojo y azul) en el centro, rodeados por barras negras arriba y abajo.</p>	 <p>Diagrama de cuadrícula nuevo con dos mosaicos verticales (rojo y azul) en el centro, rodeados por barras negras a los lados.</p>
<p>Mejora la utilización del espacio al reducir las distancias y minimizar las barras negras.</p>	 <p>Diagrama de cuadrícula antiguo con cinco mosaicos (rojo, azul, amarillo, naranja, verde) dispersos en una cuadrícula, con grandes espacios y barras negras.</p>	 <p>Diagrama de cuadrícula nuevo con cinco mosaicos (rojo, azul, amarillo, naranja, verde) compactados en una cuadrícula, reduciendo el espacio y las barras negras.</p>
<p>Agrega un nuevo indicador de "cámara apagada" para ver claramente a los participantes que no comparten video.</p>	 <p>Diagrama de cuadrícula antiguo con una barra negra que indica que la cámara está apagada.</p>	 <p>Diagrama de cuadrícula nuevo con iconos de cámara apagada (triángulo invertido) que indican que la cámara está apagada.</p>

Descripción del cambio	Antiguo	New
<p>Mejora la utilización del espacio y las proporciones cuando se usa el modo vertical.</p>		
<p>Mejora la utilización del espacio cuando se usa el modo vertical, ya que minimiza el espacio entre los participantes y reduce el efecto de bandas negras horizontales o verticales.</p>		

6 de febrero de 2024

Compatibilidad con OBS y WHIP

IVS se puede utilizar con codificadores compatibles con WHIP, como OBS, para publicar en transmisiones en tiempo real de IVS. WHIP (protocolo de ingesta WebRTC-HTTP) es un borrador de IETF desarrollado para estandarizar la ingesta de WebRTC. Consulte la nueva página sobre [compatibilidad con OBS y WHIP](#).

1 de febrero de 2024

SDK de transmisión de Amazon IVS: versión 1.14.1 para Android, versión 1.14.1 para iOS y versión 1.8.0 para web (transmisión en tiempo real)

Plataforma	Descargas y cambios
SDK de transmisión para web (versión 1.8.0)	<p>Documentación de referencia: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference</p> <ul style="list-style-type: none"> • La codificación en capas con transmisión simultánea ahora está deshabilitada de forma predeterminada. • Se ha corregido un error que provocaba que una instancia de una fase no se desconectara correctamente cuando se eliminaba una fase o cuando un participante se desconectaba del servidor. El SDK ahora emite un evento <code>STAGE_CONNECTION_STATE_CHANGED</code> con el estado <code>DISCONNECTED</code> (en lugar de <code>ERRORED</code> y luego <code>CONNECTING</code>).

Plataforma	Descargas y cambios
	<ul style="list-style-type: none">• Se ha corregido un problema por el que la publicación fallaba al actualizar la estrategia con pistas de audio o video vacías.
SDK de transmisión para Android (versión 1.14.1)	<p>Documentación de referencia: https://aws.github.io/amazon-ivs-broadcast-docs/1.14.1/android</p> <ul style="list-style-type: none">• La codificación en capas con transmisión simultánea ahora está deshabilitada de forma predeterminada.• Se ha actualizado <code>libWebRTC</code> de M108 a M119.• Se han corregido varios bloqueos para mejorar la estabilidad general.• Se ha agregado compatibilidad con la publicación en estéreo. Esto se puede habilitar a través del objeto <code>StageAudioConfiguration</code>.• Se ha corregido un error que provocaba que la actividad de los participantes quedara en negro al unirse a una sesión.• Se han actualizado las referencias de <code>libWebRTC</code> internas para evitar conflictos de símbolos cuando se incluyen otras versiones de <code>libWebRTC</code> en la misma aplicación host.

Plataforma	Descargas y cambios
SDK de transmisión para iOS (versión 1.14.1)	<p>Descarga para transmisión en tiempo real: https://broadcast.live-video.net/1.14.1/AmazonIVSBroadcast-Stages.xcframework.zip</p> <p>Documentación de referencia: https://aws.github.io/amazon-ivs-broadcast-docs/1.14.1/ios</p> <ul style="list-style-type: none"> • La codificación en capas con transmisión simultánea ahora está deshabilitada de forma predeterminada. • Se ha actualizado libWebRTC de M108 a M119. • Se han corregido varios bloqueos para mejorar la estabilidad general. • Se ha agregado compatibilidad con la publicación en estéreo. Esto se puede habilitar mediante <code>IVSLocalStageStreamAudioConfiguration</code>. • Se ha corregido un bloqueo al habilitar el modo de solo audio para otros participantes. • Se ha mejorado el TTV y se ha reducido el tamaño de los binarios.

Tamaño del SDK de transmisión: Android

Arquitectura	Tamaño comprimido	Tamaño sin comprimir
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

Tamaño del SDK de transmisión: iOS

Arquitectura	Tamaño comprimido	Tamaño sin comprimir
arm64	3,350 MB	7,790 MB

3 de enero de 2024

SDK de transmisión de Amazon IVS: versión 1.13.4 para Android, versión 1.13.4 para iOS y versión 1.7.0 para web (transmisión en tiempo real)

Plataforma	Descargas y cambios
SDK de transmisión para web (versión 1.7.0)	<p>Documentación de referencia: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference</p> <ul style="list-style-type: none"> Se ha mejorado el tiempo de reproducción de video para los suscriptores que se unen a las fases. Se ha eliminado la propiedad <code>minAudioBitrateKbps</code> (no se usaba). Se ha mejorado la recuperación de la red durante interrupciones o cambios de Internet.
SDK de transmisión para Android (versión 1.13.4)	<p>Documentación de referencia: https://aws.github.io/amazon-ivs-broadcast-docs/1.13.4/android</p> <ul style="list-style-type: none"> StageAudioConfiguration ahora permite configurar si la cancelación de eco debe estar habilitada.
SDK de transmisión para iOS (versión 1.13.4)	<p>Descarga para transmisión en tiempo real: https://broadcast.live-video.net/1.13.4/AmazonIVSBroadcast-Stages.xcframework.zip</p>

Plataforma	Descargas y cambios
	<p>Documentación de referencia: https://aws.github.io/amazon-ivs-broadcast-docs/1.13.4/ios</p> <ul style="list-style-type: none"> • En iOS, hemos mejorado el motor de audio tanto para la grabación como para la reproducción, centrándonos en la estabilidad y la capacidad de recuperación. Esto mejora la compatibilidad con los cambios de ruta durante el uso, mejora la recuperación de la batería en los casos de borde y reduce la cantidad de bloqueos del subproceso principal. • Se ha corregido un error que provocaba que el micrófono permaneciera activo incluso después de haberlo desconectado de una fase, lo que dejaba encendido el indicador de privacidad de iOS. (El SDK no procesaba el audio entrante en ese momento).

Tamaño del SDK de transmisión: Android

Arquitectura	Tamaño comprimido	Tamaño sin comprimir
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

Tamaño del SDK de transmisión: iOS

Arquitectura	Tamaño comprimido	Tamaño sin comprimir
arm64	3,45 MB	7,84 MB

7 de diciembre de 2023

Nuevas métricas de CloudWatch

Cambiamos el nombre de la métrica PacketLoss (Stage) a DownloadPacketLoss (Stage).
Publicamos métricas adicionales de CloudWatch para la transmisión en tiempo real de IVS:

- DownloadPacketLoss (Stage,Participant)
- DroppedFrames (Stage,Participant)
- SubscribeBitrate (Stage,Participant,MediaType)

Para obtener más información, consulte [Monitoreo de transmisión en tiempo real de IVS](#).

4 de diciembre de 2023

SDK de transmisión de Amazon IVS: Android 1.13.2 e iOS 1.13.2 (transmisión en tiempo real)

Plataforma	Descargas y cambios
Todos los dispositivos móviles (Android e iOS)	<ul style="list-style-type: none"> • La configuración de supresión de ruido está disponible para que los desarrolladores la habiliten o deshabiliten para su publicación.
SDK de transmisión para Android 1.13.2	Documentación de referencia: https://aws.github.io/amazon-ivs-broadcast-docs/1.13.2/android

Plataforma	Descargas y cambios
	<ul style="list-style-type: none"> Se ha mejorado el tiempo que se tarda en cargar el video (TTV) al unirse al primer escenario de una sesión.
SDK de transmisión para iOS 1.13.2	<p>Descarga para transmisión en tiempo real: https://broadcast.live-video.net/1.13.2/AmazonIVSBroadcast-Stages.xcframework.zip</p> <p>Documentación de referencia: https://aws.github.io/amazon-ivs-broadcast-docs/1.13.2/ios</p> <ul style="list-style-type: none"> No hay cambios en el SDK en tiempo real.

Tamaño del SDK de transmisión: Android

Arquitectura	Tamaño comprimido	Tamaño sin comprimir
arm64-v8a	5177 MB	13,01 MB
armeabi-v7a	4485 MB	9,045 MB
x86_64	5,352 MB	13,808 MB
x86	5547 MB	14 192 MB

Tamaño del SDK de transmisión: iOS

Arquitectura	Tamaño comprimido	Tamaño sin comprimir
arm64	3,45 MB	7,82 MB

21 de noviembre de 2023

SDK de transmisión de Amazon IVS: Android 1.13.1 (transmisión en tiempo real)

Plataforma	Descargas y cambios
SDK de transmisión para Android 1.13.1	<p>Documentación de referencia: https://aws.github.io/amazon-ivs-broadcast-docs/1.13.1/android</p> <ul style="list-style-type: none"> Se ha corregido un error que provocaba un bloqueo al salir, soltar y volver a entrar rápidamente en el mismo escenario.

Tamaño del SDK de transmisión: Android

Arquitectura	Tamaño comprimido	Tamaño sin comprimir
arm64-v8a	5177 MB	13 102 MB
armeabi-v7a	4485 MB	9046 MB
x86_64	5353 MB	13 809 MB
x86	5547 MB	14 192 MB

17 de noviembre de 2023

SDK de transmisión de Amazon IVS: Android 1.13.0 e iOS 1.13.0 (transmisión en tiempo real)

Plataforma	Descargas y cambios
Todos los dispositivos móviles (Android e iOS)	<ul style="list-style-type: none"> Optimizaciones de transmisión actualizado. Entre otras cosas, la característica “Transmis

Plataforma	Descargas y cambios
	<p>ión adaptativa: codificación por capas con transmisión simultánea” ahora requiere una suscripción explícita y solo es compatible con las versiones recientes de SDK.</p> <ul style="list-style-type: none">• Se mejoró la estabilidad de las fases al reducir los fallos poco frecuentes.• Se ha mejorado el tiempo que se tarda en cargar el vídeo (TTV) al unirse a un escenario.• Se ha mejorado la experiencia con los dispositivos Bluetooth.• Se optimizó el uso de la CPU y la memoria del SDK y se redujo el tamaño de la biblioteca.• Se agregó la clase <code>StageAudioManager</code> , que se puede usar para configurar los parámetros de captura y reproducción de audio, incluidos los ajustes preestablecidos para la comunicación de voz, la reproducción multimedia y más. Para obtener más información, consulte la nueva página, IVS Broadcast SDK: Mobile Audio Modes.• Se agregó una nueva función <code>requestQualityStats</code> para mostrar eventos de calidad estructurados a partir de las estadísticas de WebRTC.• Se agregó una nueva función para actualizar la tasa de bits del audio. Se establece en <code>LocalStageStream</code> objetos igual que en la configuración de vídeo, pero a través de un nuevo objeto de configuración de audio.

Plataforma	Descargas y cambios
SDK de transmisión para Android 1.13.0	<p>Documentación de referencia: https://aws.github.io/amazon-ivs-broadcast-docs/1.13.0/android</p> <ul style="list-style-type: none">• Todos los métodos de la interfaz <code>StageRenderer</code> ahora son opcionales.• Se ha añadido compatibilidad de <code>SurfaceView</code> con la vista previa basada en datos para mejorar el rendimiento. Los métodos existentes <code>getPreview</code> en <code>Session</code> y <code>StageStream</code> siguen devolviendo una subclase de <code>TextureView</code>, pero esto puede cambiar en una futura versión del SDK.• Si su aplicación depende específicamente de <code>TextureView</code>, puede continuar sin cambios. También puede cambiar de <code>getPreview</code> a <code>getPreviewTextureView</code> para prepararse para el eventual cambio de lo que <code>getPreview</code> devuelve como valor predeterminado.• Si su aplicación no requiere <code>TextureView</code> específicamente, le recomendamos que cambie a <code>getPreviewSurfaceView</code> para reducir el uso de CPU y memoria.• El SDK ahora implementa un nuevo tipo de vista previa denominada <code>ImagePreviewSurfaceTarget</code> que funciona con el objeto <code>Android Surface</code> proporcionado por la aplicación. No es una subclase de <code>Android View</code>, que proporciona una mayor flexibilidad.• Se corrigió el caso en el que la devolución de llamada <code>onFrame</code> para un participante

Plataforma	Descargas y cambios
	<p>remoto se llamaba en el momento incorrecto y con un tamaño incorrecto.</p> <ul style="list-style-type: none">• <code>SurfaceSource # getInputSurface</code> ahora está anotado con <code>@Nullable</code> . Debe comprobar su código antes de usarlo.• Se han añadido <code>UserId</code>, <code>attributes</code> y <code>ParticipantInfo</code> . Las propiedades <code>UserId</code> y <code>attributes</code> están integradas en el token y las aplicaciones pueden recuperarlas a través de <code>ParticipantInfo</code> siempre que un participante se una.• La captura de cámara y el procesamiento de la vista previa ahora tienen un valor predeterminado de 720 x 1280 o la resolución de publicación (la que sea mayor) es de 15 fps. Puede ajustar la resolución o los fps utilizando <code>StageVideoConfiguration # setCameraCaptureQuality</code> .• <code>IllegalArgumentException</code> que aparece al establecer las propiedades de configuración ahora incluye el valor proporcionado en el mensaje de excepción.

Plataforma	Descargas y cambios
SDK de transmisión para iOS 1.13.0	<p>Descarga para transmisión en tiempo real: https://broadcast.live-video.net/1.13.0/AmazonIVSBroadcast-Stages.xcframework.zip</p> <p>Documentación de referencia: https://aws.github.io/amazon-ivs-broadcast-docs/1.13.0/ios</p> <ul style="list-style-type: none">• Se solucionó el problema por el que el SDK no cambiaba la configuración de vídeo si la configuración de vídeo se actualizaba antes de la publicación.• Se incorporó la solución de Google para una vulnerabilidad de seguridad de LibVPX (CVE-2023-5217). (Tenga en cuenta que el SDK de Android no requirió ningún cambio por este problema).• Las aplicaciones que utilicen otras bibliotecas que incluyan <code>libWebRTC</code> ya no tendrán conflictos con el SDK de IVS Broadcast.• Todos los métodos del protocolo <code>IVSStageRenderer</code> ahora están marcados como <code>@optional</code>.• Los micrófonos y las cámaras devueltos por nuestros SDK ahora tienen un orden de clasificación garantizado, tal y como se documenta en los propios SDK.• Ahora, varias cámaras pueden tener un valor de <code>true</code> para su propiedad de <code>isDefault</code>, uno para cada posición, según lo determine el sistema operativo.• Se ha añadido <code>IVSStageAudioManager</code>, que permite un control preciso de la función subyacente <code>AVAudioSession</code> para

Plataforma	Descargas y cambios
	<p>ampliar la variedad de casos de uso de la funcionalidad de Escenarios.</p> <ul style="list-style-type: none"> • Se agregó UserId a ParticipantInfo .

Tamaño del SDK de transmisión: Android

Arquitectura	Tamaño comprimido	Tamaño sin comprimir
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

Tamaño del SDK de transmisión: iOS

Arquitectura	Tamaño comprimido	Tamaño sin comprimir
arm64	3,45 MB	7,84 MB

16 de noviembre de 2023

Grabación compuesta

Esta nueva característica permite grabar la vista compuesta de un IVS Stage en un bucket de Amazon S3. Para obtener más información, consulte:

- [Grabación compuesta](#): esta es una página nueva.
- [Introducción a transmisión de IVS en tiempo real](#): en la sección “Configurar los permisos de IAM”, añadimos los puntos de conexión S3 a la política.

- [Service Quotas de IVS](#): se agregaron cuotas de tarifas de llamada para los nuevos puntos de conexión.
- [Referencia de la API de transmisión de IVS en tiempo real](#): agregamos 4 puntos de conexión de StorageConfiguration y 7 objetos (DestinationDetail, RecordingConfiguration, S3DestinationConfiguration, S3Detail, S3StorageConfiguration, StorageConfiguration, StorageConfiguration Summary). También modificamos 3 objetos (Composition, Destination y DestinationConfiguration); esto afecta a la respuesta de GetComposition y a la solicitud y respuesta de StartComposition.

16 de noviembre de 2023

Composición del servidor

La composición del servidor de IVS permite a los clientes transferir la composición y la emisión de un escenario de IVS a un servicio gestionado por IVS. La composición del servidor y la transmisión RTMP a un canal se invocan a través de los puntos de conexión del plano de control del IVS en la región de origen del escenario. Para obtener más información, consulte:

- [Introducción a transmisión de IVS en tiempo real](#): en el apartado “Configurar los permisos de IAM”, añadimos puntos de conexión de SSC a la política.
- [Uso de Amazon EventBridge con la transmisión de IVS en tiempo real](#): añadimos nuevas métricas.
- [Composición del servidor](#): este nuevo documento incluye una descripción general e instrucciones de configuración.
- [Service Quotas \(transmisión en tiempo real\)](#): agregamos nuevos límites de tarifa de llamadas y otras cuotas.
- [Referencia de la API de transmisión en tiempo real](#): hemos añadido 8 puntos de conexión y 11 objetos (ChannelDestinationConfiguration, Composition, CompositionSummary, Destination, DestinationConfiguration, DestinationSummary, EncoderConfiguration, EncoderConfigurationSummary, EncoderConfigurationSummary, GridConfiguration, LayoutConfiguration y Video).

En la Guía del usuario de transmisión de baja latencia de IVS, consulte:

- [Habilitación de varios hosts en una transmisión de IVS](#): agregamos “Transmisión de un escenario: composición del cliente versus composición del servidor” y actualizamos “4. Transmisión del escenario”.

16 de octubre de 2023

SDK de transmisión de Amazon IVS: Web 1.6.0 (transmisión en tiempo real)

Plataforma	Descargas y cambios
SDK de transmisión web 1.6.0	<p>Documentación de referencia: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference</p> <ul style="list-style-type: none">• Tiempo de reproducción de video (TTV) mejorado.• Configuración adicional <code>maxAudioBitrate</code>, que admite hasta 128 kbps de canales de audio mono o estéreo.

12 de octubre de 2023

Nuevas métricas y datos de los participantes de CloudWatch

Hemos publicado métricas de CloudWatch para transmisión en tiempo real de IVS. Para obtener más información, consulte [Monitoreo de transmisión en tiempo real de IVS](#).

También hemos agregado seis campos al objeto de la API de participantes: `browserName`, `browserVersion`, `ispName`, `osName`, `osVersion` y `sdkVersion`. Esto afecta a la respuesta de `GetParticipant`. Consulte la [Referencia de la API de transmisión en tiempo real de IVS](#).

12 de octubre de 2023

SDK de transmisión de Amazon IVS: Android 1.12.1 (transmisión en tiempo real)

Plataforma	Descargas y cambios
SDK de transmisión para Android 1.12.1	<p>Documentación de referencia: https://aws.github.io/amazon-ivs-broadcast-docs/1.12.1/android</p> <ul style="list-style-type: none">Se ha corregido un error por el cual <code>BroadcastSession.setListener</code> provocaba un error al llamar.

Tamaño del SDK de transmisión: Android

Arquitectura	Tamaño comprimido	Tamaño sin comprimir
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 de septiembre de 2023

SDK de transmisión de Amazon IVS: Web 1.5.2 (transmisión en tiempo real)

Plataforma	Descargas y cambios
SDK de transmisión para web 1.5.2	<p>Documentación de referencia: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference</p> <ul style="list-style-type: none">Se ha corregido un error que impedía volver a publicar <code>refreshStrategy</code> cuando el estado publicado entraba en un estado <code>ERRORED</code>.

23 de agosto de 2023

SDK de transmisión de Amazon IVS: Web 1.5.1, Android 1.12.0 e iOS 1.12.0 (transmisión en tiempo real)

Plataforma	Descargas y cambios
SDK de transmisión para web 1.5.1	<p>Documentación de referencia: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference</p> <ul style="list-style-type: none">Se ha corregido un error con los tipos <code>Maybe</code> internos en TypeScript 5.Se ha agregado una mejor detección para la compatibilidad con la transmisión simultánea.Se han corregido dos condiciones de carrera con <code>refreshStrategy</code> al intentar publicar.

Plataforma	Descargas y cambios
	<ul style="list-style-type: none"> Se ha corregido una condición de carrera con <code>refreshStrategy</code> al intentar actualizar a los participantes a los que suscribirse.
Todos los dispositivos móviles (Android e iOS)	<ul style="list-style-type: none"> Se ha solucionado un problema poco frecuente por el que la acción de publicación nunca se completaba. Se mejoró la estabilidad de las fases al reducir los fallos poco frecuentes. Se ha mejorado la estabilidad de las etapas al resolver los problemas relacionados con las condiciones de la carrera que se producían al entrar o salir rápidamente. Se ha agregado un nuevo método <code>setOnFrameCallback</code> en <code>ImageDevice</code>. Esto permite la observación a medida que los fotogramas se transfieren al propio dispositivo, lo que permite obtener información sobre la relación de aspecto de las imágenes más recientes. Este método también se puede utilizar para detectar cuándo se renderiza el primer fotograma para un participante remoto en un escenario.
SDK de transmisión para Android 1.12.0	<p>Documentación de referencia: https://aws.github.io/amazon-ivs-broadcast-docs/1.12.0/android</p> <ul style="list-style-type: none"> Ahora se admite Android 9. Uso y rendimiento mejorados de la CPU.

Plataforma	Descargas y cambios
SDK de transmisión para iOS 1.12.0	<p>Descarga para transmisión en tiempo real: https://broadcast.live-video.net/1.12.0/AmazonIVSBroadcast-Stages.xcframework.zip</p> <p>Documentación de referencia: https://aws.github.io/amazon-ivs-broadcast-docs/1.12.0/ios</p> <ul style="list-style-type: none"> Se corrigió la firma de <code>IVSDeviceDiscovery.createAudioSourceWithName</code> para devolver un <code>IVSCustomAudioSource</code> en lugar de <code>IVSCustomImageSource</code>.

Tamaño del SDK de transmisión: Android

Arquitectura	Tamaño comprimido	Tamaño sin comprimir
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

Tamaño del SDK de transmisión: iOS

Arquitectura	Tamaño comprimido	Tamaño sin comprimir
arm64	5,06 MB	10,92 MB

7 de agosto de 2023

SDK de transmisión de Amazon IVS: Web 1.5.0, Android 1.11.0 y iOS 1.11.0

Plataforma	Descargas y cambios
<p>SDK de transmisión para Web 1.5.0</p>	<p>Documentación de referencia: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference</p> <ul style="list-style-type: none"> Transmisión simultánea agregada: cuando está habilitada, esta característica permite al publicador enviar capas de video de alta y baja calidad. Los suscriptores seleccionan automáticamente la calidad óptima en función de las condiciones de la red. Consulte Optimización del contenido multimedia.
<p>Todos los dispositivos móviles (Android e iOS)</p>	<p>Transmisión simultánea agregada: cuando está habilitada, esta característica permite al publicador enviar capas de video de alta y baja calidad. Los suscriptores seleccionan automáticamente la calidad óptima en función de las condiciones de la red. Consulte “Activación o desactivación de la codificación en capas con transmisión simultánea” en las Guías del SDK de transmisión para Android y iOS.</p>
<p>SDK de transmisión para Android 1.11.0</p>	<p>Documentación de referencia: https://aws.github.io/amazon-ivs-broadcast-docs/1.11.0/android</p> <ul style="list-style-type: none"> Se ha corregido un error que provocaba que la creación de muchos escenarios acabara causando un bloqueo. (El número exacto de escenarios depende del dispositivo).

Plataforma	Descargas y cambios
SDK de transmisión para iOS 1.11.0	<p>Descarga para transmisión en tiempo real: https://broadcast.live-video.net/1.11.0/AmazonIVSBroadcast-Stages.xcframework.zip</p> <p>Documentación de referencia: https://aws.github.io/amazon-ivs-broadcast-docs/1.11.0/ios</p> <ul style="list-style-type: none"> Se corrigió la firma de <code>IVSDeviceDiscovery.createAudioSourceWithName</code> para devolver <code>IVSCustomAudioSource</code> en lugar de <code>IVSCustomImageSource</code>.

Tamaño del SDK de transmisión: Android

Arquitectura	Tamaño comprimido	Tamaño sin comprimir
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

Tamaño del SDK de transmisión: iOS

Arquitectura	Tamaño comprimido	Tamaño sin comprimir
arm64	5,030 MB	10,810 MB

7 de agosto de 2023

Transmisión en tiempo real

La transmisión en tiempo real de Amazon Interactive Video Service (IVS) le permite ofrecer transmisiones en directo con una latencia inferior a 300 milisegundos desde el host hasta el espectador.

Esta versión incluye cambios importantes en la documentación. La [página de destino de la documentación de Amazon IVS](#) ahora tiene secciones diferentes para la transmisión en tiempo real y la transmisión de baja latencia. Cada sección tiene su propia guía del usuario y referencia de API. Para obtener más información sobre el documento, consulte el historial del documento (para [tiempo real](#) y [baja latencia](#)). Para la transmisión en tiempo real, comience con la [Guía del usuario de transmisión en tiempo real de IVS](#) y la [Referencia de la API de transmisión en tiempo real de IVS](#).